The USENIX Association Magazine

;login:

August 1998

volume 23 • number 4

inside:

features:

USENIX The Advanced Computing Systems Association

upcoming events

3rd USENIX Workshop on Electronic Commerce

WHEN	WHERE	WHO
August 31-Sept. 3/98	Boston, MA	**Bennet Yee,** Program Chair **Dan Geer,** Public Key Infrastructure Session Coordinator

6th Annual Tcl/Tk Conference

WHEN	WHERE	WHO *program co-chairs*
September 14-18/98	San Diego, CA	**Don Libes &** **Michael J. McLennan**

1st International System Administration and Networking (SANE) Conference

Organized by NLUUG, cosponsored by USENIX and Stichting NLnet

WHEN	WHERE	WHO *program co-chairs*
November 18-20/98	Maastricht, The Netherlands	**Edwin Kremer &** **Jan Christiaan van Winkel**

12th Systems Administration Conference (LISA '98)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO
December 6-11/98	Boston, MA	**Xev Gittler** & **Rob Kolstad,** Program Co-chairs **Phil Scarr** & **Pat Wilson,** IT Coordinators

DEADLINES

Final Papers
October 16/**98**

NordU99 - 1st Nordic EurOpen/USENIX Conference

WHEN	WHERE
February 9-12/99	Stockholm, Sweden

3rd Symposium on Operating Systems Design and Implementation

Co-sponsored by ACM SIGOPS and IEEE TCOS

WHEN	WHERE	WHO *program co-chairs*
February 22-25/99	New Orleans, LA	**Margo Seltzer & Paul Leach**

DEADLINES

Notification to Authors	Final Papers
October 13/**98**	January 6/**99**

1st Conference on Network Administration

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO *program chair*
April 7-9/99	Santa Clara, CA	**David Williamson**

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
November 6/**98**	December 1/**98**	February 23/**99**

5th Conference on Object-Oriented Technologies and Systems (COOTS)

WHEN	WHERE	WHO *program chair*
May 3-7/99	San Diego, CA	**Murthy V. Devarakonda**

DEADLINES

Extended Abstracts	Notification of Acceptance	Final Papers
November 6/**98**	Dec. 16/**98**	March 23/**99**

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 7-11/99	Monterey, CA	**Avi Rubin,** Program Chair **Clem Cole** **& John Heidemann,** IT Coordinators **Jordan Hubbard,** Freenix Track Chair

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
December 2/**98**	January 20/**99**	April 27/**99**

Eighth USENIX Security Symposium

WHEN	WHERE	WHO
August 23-26, 1999	Washington, D.C.	**Win Treese,** Program Chair **Avi Rubin,** IT Coordinator

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
March 16/**99**	April 21/**99**	July 12/**99**

2nd Conference on Domain-Specific Languages

Sponsored by USENIX in cooperation with ACM SIGPLAN and SIGSOFT

WHEN	WHERE	WHO *program chair*
October 3-6/99	Austin, TX	**Thomas Ball**

DEADLINES

Paper Submissions	Notification to Authors	Final Papers
March 22/**99**	June 2/**99**	August 24/**99**

2nd USENIX Symposium on Internet Technologies and Systems

In cooperation with the IEEE Computer Society Task Force on Internetworking (pending)

WHEN	WHERE	WHO *program chair*
October 11-14/99	Boulder, CO	**Fred Douglis**

DEADLINES

Extended Abstracts	Notification to Authors	Final Papers
April 15/**99**	May 28/**99**	August 31/**99**

For a complete list of future USENIX events, access *http://www.usenix.org/events*

contents

;login:

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) Volume 23, Number 4 (August 1998) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$40 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Editor:
Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:
Tina Darmohray <tmd@usenix.org>

Standards Report Editor:
Nick Stoughton <nick@usenix.org>

Managing Editor:
Eileen Cohen <cohen@usenix.org>

Copy Editor:
Sylvia Stein Wright

Proofreader:
Kay Keppler

Designer:
Vinje Design

Typesetter:
Festina Lente

Advertising

Linda Barnett <barnett@usenix.org>

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org>

The closing dates for submission to the next two issues of *;login:* are September 29, 1998 and December 9, 1998.

in this issue . . .

by Eileen Cohen
;login: managing editor
<cohen@usenix.org>

Volumes have been and no doubt will be written about object-oriented programming. I haven't actually put the 976-page *Object-Oriented Programming*, the first volume of Peter Salus's new four-volume *Handbook of Programming Languages*, on a scale, but the sore muscles I got lugging my autographed copy back from the USENIX Annual Technical Conference in New Orleans this June certainly attest to the weightiness of the material.

Object-oriented themes abound in this issue of *;login:*. The 4th USENIX Conference on Object Oriented Technologies and Systems took place in April; for the second year in a row we thank Irfan Pyarali for contributing his excellent summaries of the COOTS technical sessions. Joseph Hall's "Effective Perl Programming" column reveals the object-oriented features of Perl, and Glen McCluskey's and Prithvi Rao's columns explore the performance and security aspects, respectively, of Java.

In addition, you'll find plenty of purely practical guidance in this issue. The current installment of Bob Gray's series on source code UNIX for the PC – following previous columns that convinced you to set up a system and helped you to buy the hardware – steps you through loading the software. There's solid advice from our reliability guru, John Sellens, on disaster recovery planning. And several articles offer or clarify specific tools: tools for personal email management, network monitoring, text generation from large data sets, and Tcl.

On the lighter side we offer Rob Kolstad's interview with John Stewart, a guy on the startup fast track, and Rik Farrow's ruminations on user interfaces and how engineers think.

Hope you're having a great summer!

letters to the editor

Unique UIDs

While reading the summary of the conference session "A Highly Scalable Electronic Mail Service Using Open Systems" in the April issue of *;login:*, I came across a curious statement: "With 400,000 users, it is no longer possible to assign each user a unique UID in the traditional UNIX sense."

It may be true that older flavors of UNIX cannot deal with 400,000 unique UIDs, which typically allow only 64k UIDs, but it is certainly not true for most newer flavors. Most newer flavors of UNIX use 32-bit integers to hold UIDs, which enables them to recognize over four billion UIDs (two billion if you don't like negative numbers). I believe this is comfortably enough larger than 400,000 to disqualify the above statement.

Now that I've flamed this article a little, let me say that I found it generally excellent and the technical work behind it quite interesting. Not to mention that there may be serious interpretability and performance problems when using this many UIDs in practice. But that doesn't change the fact that the above statement is demonstratively incorrect.

Christopher J. Calabrese, BFR Systems
<cjc@fpk.hp.com>

Computer Attacks

I suspect that there are typos on Page 48 in the article titled "Trends in Computer Attacks" (*;login:* Special Issue on Security, May 1998). In the section listing changes that could/should be made to Cisco products, there is a list of "access-list 111" rules. The first two entries:

```
access-list 111 deny ip 0.0.0.0
        255.255.255.255 any
access-list 111 deny ip 255.255.255.255
        255.255.255.255 any
```

actually block the same thing: All IPs. These entries will stop all traffic coming into the network. I think what was meant was having the lines read:

```
access-list 111 deny ip 0.0.0.0 0.0.0.0
        any
access-list 111 deny ip 255.255.255.255
        0.0.0.0 any
```

(also acceptable is "access-list 111 deny host 0.0.0.0 any", and the same for 255.255.255.255).

The second typo that I spotted was right below the first:

```
access-list 222 permit ip <your-
        network-address> <mask> any
```

The rest of that section talks about access-list 122, so I would assume the above line should have started "access-list 122".

The second typo isn't major, but the first could be if it is simply copied into the router configuration.

Thanks.

Theo Van Dinter
<tvd@chrysalis.com>

Rik Farrow (special issue editor) replies:

You are correct in both cases. Cisco uses a one to indicate that an address bit can be ignored (a wild card), and a zero to mean that the address bit must match exactly. The access control list rules that were published were incorrect. If I hadn't been asleep during the final page proof, I would have caught that. The second typo, using access list number 222 had been corrected on an earlier version, but the mistake reappeared in the final version.

conference reports

This issue's reports focus on the **4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)**, held in Santa Fe, New Mexico, on April 27-30, 1998, and on the **7th Annual System Administration, Networking and Security Conference (SANS)**, a conference co-sponsored by SAGE, held in Monterey, California, on May 9-15, 1998.

Our thanks to the Summarizers:

For COOTS

Irfan Pyarali
<irfan@cs.wust.edu>

For SANS

Srinath Alapati
<srinath@colltech.com>

Dave Bianchi
<djb@colltech.com>

Allen Canning
<canning@colltech.com>

Charles Gagno
<charles@colltech.com>

Brian Kirouac
<bri@colltech.com>

Our thanks to the COOTS Photographer:

Joe Hoffert
<joeh@cs.wust.edu>

4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

SANTA FE, NEW MEXICO

April 27-30, 1998

Summaries by Irfan Pyarali

Once the Spanish and Mexican capital of the region, Santa Fe is like no other city on earth. At an elevation of about 7,000 feet (2,135 meters) above sea level, Santa Fe is at the point where the high desert meets the ponderosa pine and aspen forests of the Sangre de Cristo Mountains, which rise to over 12,000 feet (3,660 meters) within a half-hour's drive of town. In addition to providing a beautiful setting for outdoor activities, Santa Fe is well known as a center of art and culture, with numerous galleries, museums, opera and chamber music. Although Santa Fe is the state capital of New Mexico, it is not the largest city. With a population of approximately 70,000, it retains a charming "small town" feeling. The first USENIX C++ workshop was held here in 1987; this conference series was converted to COOTS in 1995.

When attendees were not enjoying Santa Fe, they were hearing about recent advances in tools, languages, frameworks, components, and patterns in the conference rooms of the Eldorado Hotel. Most of the technical sessions focused on the increasing demands of distributed computing, with Java stealing the show. Joseph Sventek of Hewlett-Packard Laboratories was this year's program chair. The tutorial program chair was Douglas C. Schmidt of Washington University. Session topics and tutorials focused on efficiency, distributed infrastructure and services, mobility, programming techniques, and distributed object computing models such as CORBA, DCOM/MTS, Java RMI, and Java Beans.

WELCOME NOTE

Joseph Sventek opened the conference with a welcome note to the 220 attendees. He was pleased to announce that 56 papers were submitted for review this year; 18 papers were accepted to the conference, of which the top five would appear in the *Distributed Systems Engineering Journal*. Joe thanked all the USENIX staff and conference organizers for their hard work in putting this event together. He then awarded Prashant Jain, late of Washington University and now at Fujitsu, the best student paper award for "Design and Performance of MedJava." Joe also promised to raffle off books that were on exhibition at the conference as a bribe so that attendees would stay until the end of the conference.

KEYNOTE ADDRESS

Rick Rashid

Rick Rashid, late of CMU/Mach and now vice president of Microsoft Research, gave this year's keynote address, entitled "The Shape of Things to Come." He described the changing role of computers from monolithic, immobile, number-crunching, isolated machines to small, mobile, interactive, internetworked devices that will be used for communication, collaboration, and distance learning.

Rick presented various examples of the shape of things to come: (1) virtual meetings from the comfort of your home, (2) natural language processing, including speech recognition and synthetic

voices, (3) gesture recognition, (4) virtual worlds for commerce and entertainment, (5) office assistants and answer wizards, and (6) implicit and intelligent help systems. Rick asserts that these kinds of technologies are not far away, pointing out that work on them is under way. In fact, some of these technologies will be supported by the 59 million lines of code in Windows NT 5.0.

Rick acknowledged that applications in the future will be complex and hard to develop. To facilitate the production of these complex applications, developers must rely on underlying middleware frameworks to provide distribution, replication, and reliability. The infrastructure of these systems will support self-describing objects, dynamic monitoring, interceptors, automated distribution, profile-based analysis, and virtual environments through clustering and replication.

Session: Efficiency

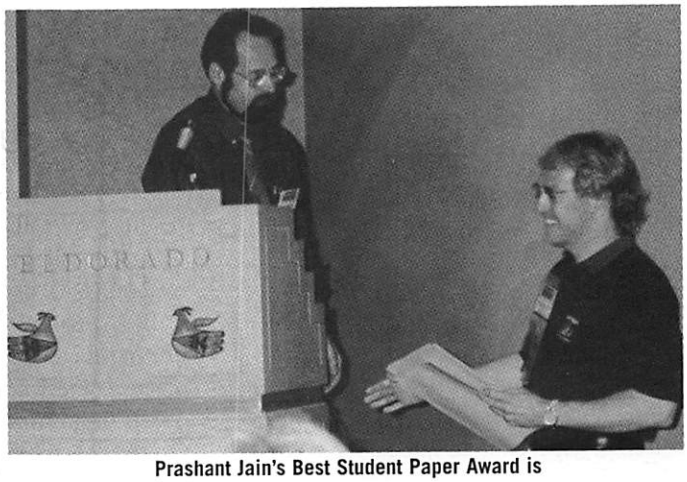
The first session of the conference was chaired by Murthy Devarakonda from IBM T.J. Watson Research Center. As distributed systems become ubiquitous and mature, increasing focus is being placed on the efficiency of these systems.

Quality of Service (QoS) Specification in Distributed Object Systems

Svend Frolund and Jari Koistinen, Hewlett-Packard Laboratories

Jari Koistinen started by pointing out that traditional distributed systems do not address QoS aspects, such as reliability, security, and performance. QoS requirements can significantly affect the design of components in the system. Jari introduced QML (QoS Modeling Language), which captures QoS properties as part of the system design.

In addition, Jari explained that some components in a distributed system are used explicitly by clients, whereas other components are used implicitly by system components that are not directly under the QoS contract specified by the client. Distributed components must therefore be QoS-aware and QoS-enabled. The infrastructure must provide negotiations, trading, and monitoring of QoS parameters. QoS-based trading services can be used to find appropriate services via dynamic matching of clients and servers.



Prashant Jain's Best Student Paper Award is accepted by coauthor Doug Schmidt

Efficient Implementation of Java Remote Method Invocation (RMI)

Vijaykumar Krishnaswamy, Dan Walther, Sumeer Bhola, Ethendranath Bommaiah, George Riley, Brad Topol, and Mustaque Ahamad, Georgia Institute of Technology

Interactive distributed applications require fast response and low latency. However, applications written in Java RMI have not met these requirements. Vijaykumar Krishnaswamy described the optimizations that make RMI more feasible for these kinds of applications. These optimizations are based on transparent caching of server objects at clients when the object changes state only infrequently. In this case, invocations can be local and do not require network communications. However, caching does require consistency protocols to make the cached objects consistent when the state of the object changes.

Vijaykumar then described experimental results that show how the caching system behaves in various client/server configurations. He showed that, as the network latency between the client and the server increases, caching pays off. However, when the cache is invalidated often, the performance is severely degraded. His approach did require the object implementor to tag operations on the object that did change the state of the object. He concluded that the RMI framework can be easily extended to support caching of objects. Moreover, point-to-point messaging between the client and server could be easily changed to employ group communication using a multicast protocol.

The Design and Performance of MedJava

Prashant Jain, Seth Widoff, and Douglas C. Schmidt, Washington University

Prashant Jain, who was at Washington University when he wrote this paper and is now at Fujitsu, began by outlining the goals of an Electronic Medical Imaging System (EMIS). He described the challenges of designing and developing an EMIS that must be reusable, efficient, reliable, flexible, and scalable. He justified the use of Java for this project by pointing out that Java is object oriented,

portable, distributed, and secure, with built-in support for threading and networking. In addition, Java has many standard libraries, and numerous Java-enabled browsers are freely available.

The EMIS Prashant developed was designed so the server contained the image repository and filter repository, while the client contained a graphical user interface, a URL locator, an image processor, a component for uploading and downloading of images, and a filter configurator. Filters can be used to view different aspects of an image, for example, to sharpen, enlarge, and edge enhancement.

After this overview, Prashant compared the image processing based on Java with that of an image-processing toolkit based on C. In some cases, the Java-based image-processing toolkits outperformed the C-based toolkits. In general, however, the C-based image-processing toolkit outperformed the Java version due to the statically compiled/optimized nature of C applications. In conclusion, Java performed surprisingly well for image processing and network transport.

Session: Distributed Infrastructure

As Rick Rashid had pointed out in the keynote address, complicated applications of the future will demand lots of features from the underlying infrastructure, such as distribution, replication, and security. This session, chaired by Steve Vinoski of IONA Technologies, focused on distributed infrastructures.

Dynamic Management of CORBA Trader Federation

Djamel Belaid, Nicolas Provenzano, and Chantal Taconet, Institut National Des Telecommunications

The authors' work on the CORBA trading service, presented by Chantal Taconet, was an extension of the CORBA specification. Session chair Steve Vinoski promptly criticized COOTS program chair Joe Sventek, who was largely responsible for devising the CORBA trading service, for providing an incomplete specification.

The trading service helps clients find the "right" object in a distributed system. The right object is typically found based on properties of interest to the client. In CORBA, objects are located by services like naming or trading. In the trading service, object implementors register object properties with the trading service. On the other side, the client specifies the properties it is interested in. The trading service is responsible for finding the appropriate matches for the client. Traders are usually federated to provide distributed lookup services to clients.

In the CORBA specification, each trader has a local view of the trader graph (i.e., it knows only about its neighbors). The default CORBA trader has the following drawbacks: (1) no concept of nearest object to client, (2) no provision for trader failures, and (3) manual establishment of trader links. Chantal then described the Dynamic Federation Graph (DFG), where the graph is globally known in the entire system, there are no cycles in the graph, and communication load is dynamically and automatically distributed among the traders. Each trader is enhanced to support a DFG. Traders use a different management interface to keep the DFG information consistent. These extensions are transparent to the exporters and the importers.

Filterfresh: Hot Replication of Java RMI Server Objects

Arash Baratloo, New York University; P. Emerald Chung, Yennun Huang, Sampath Rangarajan, and Shalini Yajnik, Lucent Technologies, Bell Laboratories

The primary goal of the Filterfresh project was to provide high availability of Java services, protect the clients from server failures, and transparently replicate services and data. Filterfresh used a fault tolerance approach based on process groups, such that the group members have a consistent view of the state of the system. The system was developed using UDP, and all events in the system are reliable and totally ordered. Experiments showed that under certain conditions local RMI was slower than remote RMI. It also showed that Java threading and synchronization were the leading cause of bottlenecks in the system.

COMERA: COM Extensible Remoting Architecture

Yi-Min Wang, Microsoft Research; Woei-Jyh Lee, New York University

Yi-Min Wang presented this work. The current COM architecture allows semi-custom marshalling, custom marshalling, and RPC transport replacement. However, most of the customization can be done only at a fairly coarse-grain level, which allows only wholesale changes to be made. COMERA allows fine-grain customization of COM middleware at various different granularities and abstraction levels. These customizations allow multi-connection channels that can be used to implement replication. These customizations can also be used to implement object failover. However, there are many open issues about security, recursive extensibility, and manageability of such a system.

Session: Distributed Services

This session was chaired by Rajendra Raj from Morgan Stanley.

Java Transactions for the Internet

M.C. Little and S.K. Shrivastava, University of Newcastle

The Web frequently suffers from failures that affect the performance and consistency of applications running over it. With the advent of Java, it is now possible to include thin clients (such as browsers) in transactions to provide fault tolerance and consistency in the presence of failures.

Little and Shrivastava have developed a transactional toolkit for Java called JTSArjuna. JTSArjuna provides persistence, concurrency, crash recovery, and object replication. In addition, JTSArjuna provides distributed transactions and a high-level API that glues together the low-level details for managing an Object Transaction Service (OTS). The configurable hierarchy of classes in JTSArjuna provides flexibility to the clients in transactional domains.

Secure Delegation for Distributed Object Environments

Nataraj Nagaratnam, Syracuse Univeristy; Doug Lea, State University of New York, Oswego

Secure delegation occurs when one object authorizes another object to access services and resources on its behalf. The Secure Delegation Model (SDM) extends current Java security features to support secure remote method invocation that may involve chains of delegated calls across distributed objects. Security policies may be controlled explicitly in application code or implicitly via administrative tools.

COBEA: A CORBA-Based Event Architecture

Chaoying Ma and Jean Bacon, University of Cambridge

Chaoying Ma started by describing the concept of events in distributed systems. Events contain descriptions of changes in the system and can be used to model asynchronous communications where the sender and the receiver can to be decoupled. She also described how event channels can be used to architect pull- and push-based systems. COBEA provides parameterized filtering, reliability, and security. COBEA also has a language to specify composite events. Chaoying concluded by providing some performance results and describing her initial experiences in developing COBEA.

Session: Experience Papers

This session was chaired by Douglas C. Schmidt from Washington University. Perhaps as a sign of the times, all the papers covered techniques for effectively using Java in real, large-scale projects.

Objects and Concurrency in Triveni: A Telecommunication Case Study in Java

Christopher Colby, Loyola University; Lalita Jategaonkar Jagadeesan, Lucent Technologies, Bell Laboratories; Radha Jagadeesan and Konstantin Laufer, Loyola University; Carlos Puchol, Lucent Technologies, Bell Laboratories

Triveni is language-independent architecture that supports systems based on finite state machines. The Triveni framework supports concurrent programming with threads and events. Programs can be freely combined using Triveni combinators. Konstantin Laufer's presentation focused on realizing Triveni in Java. A Triveni program is usually a combination of an Observable, an Observer, and a Runnable. The framework has built-in support for composing parallel tasks, event multicast, and event ordering. The next version of Triveni will include support for distributed programming in the form of Java RMI. Future work will also include mobility and support for exception and priorities.

An Object-Oriented Framework for Distributed Computational Simulation of Aerospace Propulsion Systems

John A. Reed and Abdollah A. Afjeh, University of Toledo

Designing and developing new aerospace propulsion systems is time-consuming and expensive. Onyx is a Java-based object-oriented application framework for aerospace propulsion system simulation that supports finite element analysis and computational fluid dynamics. For computationally intensive analysis, Onyx components can be distributed across a heterogeneous system. Complicated propulsion components are designed using the composite pattern. Multi-dimensional design supports increasing levels of details for the simulation.

Building a Scalable and Efficient Component Oriented System using CORBA – Active Badge System Case Study

Jakub Szymaszek, Andrzej Uszok, and Krzysztof Zielinski, University of Mining and Metallurgy, Kracow

Andrzej Uszok presented this paper. The Active Badge next generation (ABng) project is a localization system for an office environment. Used to control office and home appliances, ABng provides location and hardware abstractions, event filtering, and security. Event notifications in the system are based on the Observer pattern. The system is written in Java and CORBA.

Session: Mobility

This session was chaired by Doug Lea from the State University of New York (SUNY), Oswego.

Mobile Objects and Agents (MOA)

Dejan S. Milojicicc, William LaForge, and Deepika Chauhan, The Open Group Research Institute

MOA was designed to support migration, communication, and control of agents. It was implemented on top of the Java Virtual Machine, without any modifications to it. The initial project goals were to support communication across agent migration as a means for collaborative work and to provide extensive resource control as a basic support for countering denial of service attacks.

In the course of the MOA project, two further goals were added: (1) compliance with the Java Beans component model that provides for additional configurability and customization of agent system and agent applications and (2) interoperability that allows cooperation with other agent systems. The MOA architecture was analyzed, with particular attention being paid to the support for mobility, naming and locating, communication, and resource management. Object and component models of MOA were discussed and some implementation details described.

Programming Network Components Using NetPebbles: An Early Report

Ajay Mohindra, Apratim Purakayastha, Deborra Zukowski, and Murthy Devarakonda, IBM T.J. Watson Research Center

Developers of networkcentric applications face a number of challenges, including distributed program design, efficient remote object access, software reuse, and program deployment issues. This level of complexity hinders the developer's ability to focus on the application logic. NetPebbles removes this complexity from the developer through a network component-based scripting environment where remote object access and program deployment are transparent to the developer.

In the NetPebbles programming model, developers select network components from a distributed catalog and then write a script invoking component methods as if the components were local. When the script is launched, the Netpebbles runtime determines the component sites in the network and transparently moves the script as needed. Using three simple examples with different data flow patterns, the NetPebbles approach was shown to be superior to the traditional client/server systems and mobile agent technologies because a scripting language is easy to use, it requires less code, and the distributed systems complexity is hidden from the programmer.

The Architecture of a Distributed Virtual Worlds System

Manny Vellon, Kirk Marple, Don Mitchell, and Steven Drucker, Microsoft Research

The Virtual World project provides an object model that facilitates the development of shared virtual environments. The project is implemented on top of COM and OLE Automation and allows access from active scripting-enabled languages. The platform provides features that handle client/server computing, persistent state management, security, and ease of development.

Session: Programming Techniques

The final session was chaired by Jennifer Hamilton from Microsoft Corporation.

Execution Patterns in Object-Oriented Visualization

Wim De Pauw, David Lorenz, John Vlissides, and Mark Wegman, IBM T.J. Watson Research Center

Execution patterns are a new metaphor for visualizing execution traces of object-oriented programs. An execution pattern lets a programmer visualize and explore a program's execution at varied levels of abstraction. The view employs visual, navigational, and analytical techniques that accommodate lengthy, realworld traces. Execution patterns enhance

object-oriented visualization in three ways: (1) object communications can be visualized, (2) similar execution patterns can be generalized, and (3) it provides a foothold for characterizing system complexity. By classifying repetitive behavior automatically into high-order execution patterns, they drastically reduce the information a programmer must assimilate, with little loss of insight.

IBDL: A Language for Interface Behavior Specification and Testing

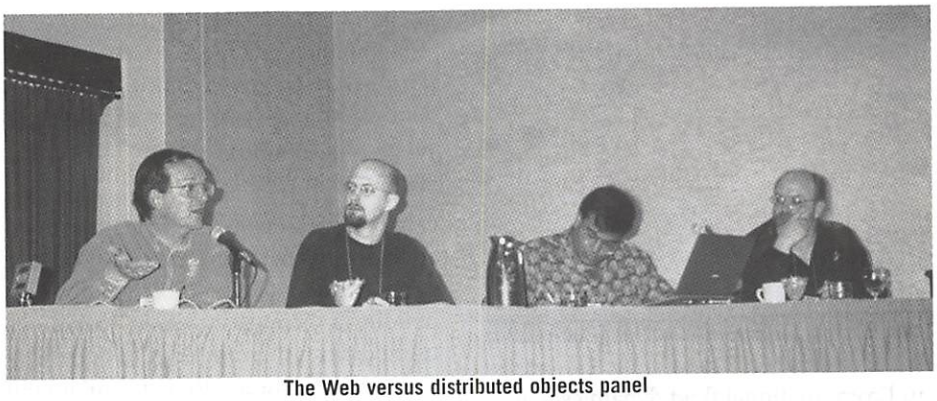
Sreenivasa Viswanadha and Deepak Kapur, State University of New York, Albany

Interface Behavior Description Language (IBDL) is a methodology and language for specifying behaviors of interfaces for object-oriented systems and is based on the message-passing paradigm. Signatures of messages are enhanced to include semantic information, expressing behavior clients can expect from a server. Formulas are given to distinguish normal termination from abnormal termination of a message using the return values and exceptions to reflect whether the precondition associated with the message is satisfied or not. State changes caused by a message invocation are specified by explicitly enumerating subsequent messages that a message invocation enables and/or disables, by establishing or violating their preconditions. Special operators on sequences of messages are defined to specify such semantic information.

Compile Time Symbolic Derivation with C++ Templates

Joseph Gil, IBM T.J. Watson Research Center; Zvi Gutterman, Technion Israel Institute of Technology

C++ templates are already recognized as a powerful linguistic mechanism whose usefulness transcends the realization of traditional generic containers. In the same venue, this presentation reported on a somewhat surprising application of templates for computing the symbolic derivative of expression. Specifically, they described a software package based on templates, called SEMT, which enables the programmer to create symbolic expressions, substitute variables in them, and compute their derivatives. SEMT is unique in that these manipulations are all done at compile time. In other words, SEMT effectively coerces the compiler to do symbolic computation as part of the compilation process. Beyond the theoretical interest, SEMT can be practically applied in the efficient, generic, and easy-to-use implementation of many numerical algorithms.



The Web versus distributed objects panel

Panel: Web Versus Distributed Objects

The panel of this year's COOTS conference was Web versus distributed objects, moderated by Ken Arnold from Sun Microsystems. The panelists were Rohit Khare from the University of California, Irvine, Joe Kiniry from the California Institute of Technology, and Jim Waldo from Sun Microsystems.

Ken began with a history of computing starting all the way back to punch cards, to object-oriented computing, and then jumping to the Web. Jim observed that the Web is full of objects that need to move around. As these objects move, they must move around with their code. This means that there was need for portable object code, safety and verification, user and code identity, and authentication. At this point, no one in the audience could tell that Jim was trying to promote Java.

The Web has always been easy to work with and use and very inexpensive to develop on. Even though the Web namespace is crummy and the protocols used are inefficient, it works. The common consensus was: just increase the network's bandwidth and everything will be OK. The conclusions from the panel were that the Web should be used for what it is good for (publishing information, user interface), and object systems should be used for more structured information exchange such as trading and mobility.

CLOSING REMARKS

Joe Sventek concluded the proceedings by thanking the attendees, the program committee, and the USENIX staff. He then introduced Murthy Devarakonda, who is the program chair for COOTS '99. Overall, COOTS '98 was a great conference with lots of novel and interesting papers and presentations. We look forward to seeing you next May in San Diego.

7th Annual System Administration, Networking and Security Conference (SANS 1998)

MONTEREY, CALIFORNIA

May 9-15, 1998

UC Davis Vulnerabilities Database

Matt Bishop, University of California, Davis

Summary by Srinath Alapati

Matt Bishop maintains the Vulnerabilities Database at the University of California, Davis. The database is a repository of information for projects in the security lab at the University of California, Davis. It is used to develop and test classification schemes, to help predict where vulnerabilities can be found, and is also being used as a guide to writing better programs.

The objectives of the Vulnerabilities Database Project, which began in 1993 are:

- learn why vulnerabilities occur
- learn how to predict and prevent them
- learn how to handle exploitation
- provide a history so future designers and implementors can learn from past mistakes and not repeat them

The latest version (version 2) of the Vulnerabilities Database is in SGML format. Current fields of the database include: V-description – vulnerability name, short and long description, how to detect a vulnerability and fix it, attack description, how to detect attack, genealogy – who reported it, when, and where. The database is on an isolated network. Modifications to the database are performed by a student or Matt Bishop. Everyone has access to the nonsensitive portion of the database. Submitters of the vulnerabilities can request to keep the portions of the vulnerabilities in confidence. Sensitive portions of the database, such as the attack scripts, are not available to the public.

Current work involves classifying vulnerabilities, which, in turn, helps to construct better detection tools rapidly, building a thesaurus to allow people using different terminology to correlate data and helping people write secure programs.

Plans for the future include creating a library of tested attack tools, upgrading and improving the existing isolated network where the database is, and building X windows and WWW interface.

Matt Bishop closed his talk by quoting Zymurgy's first law of evolving system dynamics: "When you open a can of worms, the only way to recan them is to use a large can."

The Vulnerabilities Database can be accessed at <http://seclab.cs.ucdavis.edu>. Matt Bishop can be reached at <bishop@cs.ucdavis.edu>.

Windows NT Optimization and Tuning

Mark T. Edmead, MTE Software

Summary by Srinath Alapati

Mark T. Edmead is the president of MTE Software, a Microsoft solution provider, specializing in Microsoft BackOffice products.

The presentation gave a good introduction to the Windows NT operating system:

- NT's layered architecture – kernel mode and user mode layers
- NT filesystem
- NT system resources, such as memory, processor, disk subsystem, and network subsystem.

The presentation then shifted to the "art of optimization." As Mark Edmead puts it, one will find that fixing one bottleneck brings out another. He adds that optimization is a never-ending job. However, one should realize the role of the server before attempting to optimize it.

The session explained tools such as Performance Monitor, Disk Defrag, and resources such as memory, CPU, hard disk, server, workstation, and network.

The Performance Monitor tool is available in Windows NT Workstation and Server. Windows NT's built-in core objects such as cache, memory, paging file, process, processor, system, and thread can be monitored by the Performance Monitor. These objects can be measured by various types of counters and instances.

Memory resource

Memory allocation is performed by the Virtual Memory Manager (VMM). VMM uses a page file that resides on the disk; thus memory is tied to disk performance. The number of page faults is a concern because higher numbers of page faults engender worse performance. Usage of memory counters such as "available bytes" and "committed bytes," WinMSD memory dialog, and task manager to find memory hungry applications is strongly encouraged. To prevent memory bottlenecks, take preventive actions such as adding more RAM, spreading the page file across multiple physical disks and controllers, turning off unnecessary NT services and device drivers, and disabling BIOS shadowing.

CPU Resource

CPU utilization depends on the architecture of the processor itself, the speed of the processor, the number of processors, and distributed processing. Important objects to look for are %Processor time, interrupts/sec, and system calls/sec. CPU bottlenecks can be prevented by upgrading to a faster CPU, adding more processors, and scheduling CPU intensive operations during off-peak hours. Windows

NT's Task Manager can be used to monitor CPU activity.

Hard Disk Resource

Disk technologies (IDE vs. SCSI), disk controller, and the types of filesystems (FAT vs. NTFS) play an important role in hard disk performance. Performance-related objects to monitor are % Disk time and current disk queue length. Run `diskperf -y` to turn on disk performance monitoring.

Other things that affect disk performance are disk fragmentation, head and disk speed, random or sequential access, and the seek time. Diskeeper, which will be built into NT5.0, is recommended for defragmenting the disks. Some preventive action items include using NTFS for filesystems greater than 400MB and using faster I/O bus structure (PCI over ISA).

Server and Workstation Performance

The default for server performance is Maximize Throughput for File Sharing. Server performance can be increased by increasing the server service's thread priorities, limiting the number of simultaneous logons, and disconnecting nonactive users more frequently. The workstation service handles the request for connection to a server service. Be sure the network binding order is set to the most frequently used transport protocol. The selection of protocols depends on your network. Two recommendations are use the minimum number of protocols and avoid NETBEUI when possible.

Network Resource

Types of traffic on the network, DHCP, WINS, logon requests, and directory replication are a few things to be considered when optimizing network resource.

Mark T. Edmead can be reached at <mark@mtesoft.com>.

Dealing with Spam

Rob Kolstad

Summary by Dave Bianchi

Rob gave an entertaining talk about spam (unsolicited commercial email). Spam is happening because it is very inexpensive and therefore tempting for entrepreneurs. The problem has grown to the point that, on a given day, as much as 50% of a typical email account is unsolicited advertising. Spam costs very little to send, but costs the rest of us in lost resources and productivity and can expose us to legal liabilities.

Rob talked about ways to combat spam, like personal email filters and email gateway filters. He mentioned some commercial spam filter products from companies like TIS and BSDI. He also described some rules to use to write your own spam filter:

- Don't act as a relay for other domains.
- Reject known spamming IP addresses and domains.
- Reject improper addresses in SMTP headers.
- Reject headers that do not conform to RFC-822.
- Reject erroneous, forged, or extraneous headers.
- Reject invalid or unresolvable domain names.
- Reject bad IP addresses or time/timezones.

Finally, Rob covered some `sendmail.cf` changes to implement some of these filtering rules.

Information about Rob's white paper on spam (this talk) and the BSDI spam filter product are at <http://www.bsdi.com/>.

Big Brother – Monitoring Systems and Networks Without SNMP

Sean MacGuire, The MacLawran Group Inc.

Summary by Dave Bianchi

Sean gave us an introduction to Big Brother (BB), a simple and lightweight system and network-monitoring package. BB was written (mostly over a weekend) to avoid spending over $300K on a commercial package that did the same thing.

BB provides a Web-based status-reporting system and supports paging and email for problem reporting. Sean emphasized that, for the Web status page, "green is good, red is bad," and that the Web interface is useful for everyone from sysadmins and help desk personnel to users and managers.

Sean went into detail on the structure of the package. BB consists of a set of Bourne shell scripts and a couple of C programs. The C programs make up a client-server pair of programs (using TCP port 1984) that are used to transfer status data from each BB client to the BB server. Each client sends information about disk use, CPU load, important processes, and log messages. In addition, network services like FTP, http, pop3, and smtp are tested. Every five minutes, the Web display is updated.

Big Brother has been released since October of 1996 and is available from <http://www.iti.qc.ca/iti/users/sean/bb-dnld>. Future work includes NT client and server support, providing better logging and historical system information, as well as enhanced paging functionality (by affected area, time of day, and day of the week).

Techniques for Managing DNS Files in a Large Environment

Justin Collins, Sterling Software Inc. at NASA Ames Research Center

Summary by Dave Bianchi

Justin talked about the tools that he has written to manage the DNS data at the Numerical Aerospace Simulation Facility (NAS) at NASA Ames.

Justin first described the old system. They currently use a system of manually edited data files controlled by RCS and generate hosts files (`hosts`, `hosts.equiv`, `hosts.lpd`) using a combination of Perl, Bourne shell, and C-shell. The hosts files are distributed by using `rdist` over `ssh` and `cron` jobs. The old system was not flexible or expandable and didn't provide any sanity checks for data.

Justin discussed the new system requirements. The new system must retain all current functionality, but add support for new resource record types (IPv6), support multiple domains without source code modifications, improve sanity checks for data, and provide a user interface that is easy to use.

The new system was built using a Sybase SQL server and Perl. Perl modules DBD and DBI provide the database interface, and the Tk module provides an easy-to-use X application interface. DBD and DBI allow the use of any database backend.

This project is still a work in progress. Future directions include support for BIND v8 configuration files and enhanced reporting features. Updates on the progress of this project may be found at:
<http://science.nas.nasa.gov/Groups/LAN/Projects/current/DNSms/>.

Epasswd – Solving the Heterogeneous passwd Program Problem

Eric Davis, Sterling Software Inc. at NASA Ames Research Center

Summary by Dave Bianchi

Eric described `Epasswd`, a new proactive password program. He first spoke about using `Crack` and the characteristics of passwords. He suggested that a minimum password length of eight characters was much better than the normal minimum of six. He discussed password requirements:

- Don't use a dictionary word.
- Don't choose a password similar to the old one.
- Don't choose a password containing any variation of the login name.
- Don't choose all lower- or all uppercase characters.
- Use a minimum of six characters.
- Use a mixture of lowercase, uppercase, numeric, and special characters

Eric then mentioned other proactive password tools, like `passwd+` and `npasswd`, and why these tools didn't meet their needs (like password aging and support for shadow passwords).

`Epasswd` is written in C++ and uses compile-time configuration options. It has many command line options to deal with password aging. One interesting option (`-c`) allows the setting of an encrypted password from the command line. It is available at
<http://www.nas.nasa.gov/~edavis/epasswd/>.

`Epasswd` currently works on Solaris, SunOS, IRIX, ConvexOS, and BSD operating systems. Future work will include porting to more systems, hooks for `CrackLib`, activity logging, and keeping a password history.

A Highly Integrated and Automated Network Environment

Chris Calabrese, BFR Systems

Summary by Dave Bianchi

Chris talked about his experience in centralizing the network management of systems using DHCP. The goal was to develop tools to make changes centrally, generate DHCP configuration files automatically, and report on problems like inactive hosts and duplicate IP addresses.

One unique aspect of this environment was that VLANs were used and a single switched port could be assigned to a given subnet based upon the MAC address of the system on that switched port. This meant that a system could be moved from one part of the building to another and keep the same IP address.

A prime motivation for developing these tools was to simplify a move to a new building. By converting all systems to use DHCP before the move (and by having tools to manage IP addressing), they could eliminate most of the difficulties with moving systems. As systems were moved to the new building, they were assigned new IP addresses with no manual reconfiguration needed.

The tools that were written used ARP caches, ping, and SNMP to populate the system database from which the DHCP configuration files were generated. Over time, the collected data was used to quantify subnet populations and identify unused IP addresses. Web-based status reports were generated by the tools.

Chris talked about the problems that were encountered with conversion to DHCP/BOOTP, including broken clients and clients requiring conversion scripts. The end result was a smooth move to the new building.

The tools mentioned in this talk are not publicly available.

Turning Off Sendmail Forever

Wietse Venema, IBM T.J. Watson Research Center

Summary by Dave Bianchi

Wietse spoke about VMailer, a secure replacement for sendmail. He described the way sendmail works and mentioned all the CERT advisories related to it.

Wietse's goals for VMailer are:

- wide deployment by giving it away
- compatibility: make transition easy
- performance: faster than the competition
- security: no root shells for random strangers
- flexibility: C isn't an acceptable scripting language
- reliability: behave rationally under stress
- example for a book being written with Dan Farmer

He discussed the challenges of implementing UNIX mail, including network protocols, concurrent mail database access, mail address parsing, rewriting and routing, and queue management. He also described issues concerning spam and relay control.

Wietse contrasted the monolithic mailer model of sendmail to the multiple-layer model of VMailer. In the case of sendmail, the entire mail system runs at the highest level of privilege, and one vulnerability can compromise the entire host. VMailer uses a partitioned architecture with most programs running in a chroot/low-privilege jail, and there is no trust in queue files or in IPC messages.

VMailer gets rid of security problems by eliminating the following:

- set-uid programs
- /tmp race conditions
- use of remote data in shell variables or shell commands
- fixed-length string buffers
- unbounded strings

VMailer achieves sendmail compatibility by supporting the following:

- `/etc/aliases` and NIS aliases
- `/var/spool/mail/user`, `/var/mail/user`, `user.lock`
- `$HOME/.forward`, `:include:/file/name`
- delivery to `/file/name` and `"|command"`

But it doesn't use a `sendmail.cf` file. Address rewriting and email routing involves table-driven operations providing:

- canonical: substitute `user/address/domain` (sendmail rule S3)
- virtual: redirect `user/address/domain` (sendmail rule S0)
- transports: route any domain to `relayhost`
- relocated: bounce text per local recipient
- aliases: redirect or expand local recipient

In version 1, custom address rewriting isn't supported.

On the topic of spam, Wietse talked about VMailer support for basic spam control:

- relay control
- SMTP client blacklist
- sender domain blacklist
- sender domain DNS lookup

On the TODO list is pattern-based filters for local policies.

Wietse described the queue and connection management features of VMailer. These features avoid some of the common mailer problems like inaccessible remote hosts and the deluge of mail deliveries that can occur when a host comes back up. He mentioned that the BSD fast filesystem is the limiting factor in email resources.

Finally, Wietse talked about the current status of VMailer and future plans. January 1998 started the closed alpha release. Public beta release is expected to be sometime in the second quarter of 1998. VMailer will run on most UNIX systems, but will not run under Windows. More information about VMailer can be found at <http://www.vmailer.org/>.

Making the Most of Your Opportunities with Management

Stephen Northcutt, Naval Surface Warfare Center

Summary by Dave Bianchi

Stephen gave an entertaining and well-attended talk about the ways that technical people can better communicate with management. He also mentioned that he has made at least once all of the mistakes described in this talk.

Communication

Write things down. Be willing to say "I don't know." When speaking, have a point. Use a visual aid to help convey your point. Email is not the only communications medium. If you read something in email that troubles you, use the phone to get it straightened out. Email response should be the same order of magnitude as the original message. Email provides a written audit trail. Use the Web to publish information, policy, and recommendations. Collect data and turn them into charts and graphs; publish them on the Web. Meetings should have an agenda; take notes and leave the meeting with SMART (specific, measurable, achievable, realistic, time-based) objectives.

Power

Management hires and fires, evaluates, and has power over technical people.

Never misuse the knowledge that you have. Work to minimize the loss to your employer should you leave. Build backup systems two levels deep. Share your skills; teach others. If others don't understand you, you're the one who is wrong.

Conflict

Change breeds resistance. Avoid forcing managers to resolve issues between technical groups. Don't fight your enemies; retire them. Be persistent. Use a third party to influence detractors.

Reaction

Incidents can be catalysts to get management moving. Be careful what you ask for; you might get more than you want.

Persistence

Plant early; water often. Ideas that fail today may bloom six months from now. Stay focused. Most important, never compromise your integrity; you may not be able to rebuild it.

In this talk, Stephen provided us with some good ideas about how to improve our ability to communicate and make positive changes to our organizations.

Mission Critical System Management

Yuval Lirov, Lehman Brothers

Summary by Dave Bianchi

Mr. Lirov spoke about information technology on Wall Street. He started by talking about the complexity of IT on Wall Street: technology services growing at 16% per year, large investments in mainframes and UNIX, IT turnover reaching 30%, and massive application dependency.

He outlined the goal of providing high-quality service at a low cost and emphasized the need for measurement.

He discussed the accountability metrics used, including "subjective customer perception analysis" (what do our customers perceive to be the quality of our service). Objective metrics cover availability, workload, and cost reduction. Proactive support measures like Web-based reporting of system availability and status of reported problems give perception of increased reliability.

As an example, the fixed income portfolio management tools were described as steps in dealing with the realtime nature of the Wall Street environment.

Finally, Mr. Lirov talked about the look of IT on Wall Street in ten years, based on the strategies used thus far. He sees a large investment in mainframes, UNIX, and NT. IT will be outsourced, and technology will continue to evolve at a rapid rate.

Managing User Accounts in a Distributed Network with High Turnover Rates

J. Carlson, F. Hall, S. Hu, and S. Lin, Division of Engineering Computing Services, Michigan State University

Summary by Allen Canning

The main focus of this talk was to discuss and explain the usage of DECS Account Activation System. The requirements were as follows: It had to be multiuser and multiclient and use RPC.

The server uses NIS+, so there had to be two components: a client component and a server component. The client sets up the execution environment. This is a shell script that runs when a user logs in to the system. The shell script detects what type of login (remote or console) and then executes the corresponding RPC program. The purpose of the client is to authenticate the information entered by the user against the information in the registrar's database. Once authenticated, the client sends the information to the server to do the work. Any sensitive data are sent to the server encrypted.

The server double-checks the authentication and then adds the user information to the NIS+ tables. The actual data manipulation is done by a Perl subroutine. The directory and fileserver routine, another part of the server, creates the home directory, copies initialization files, sets the correct ownership, and assigns quotas.

Their implementation has been quite successful and requires very little intervention by a system administrator.

See <http://www.egr.msu.edu/development> for more information.

Using Priv

Shaun Welch, @Home Networks

Summary by Allen Canning

This talk was about the program `Priv`. `Priv` is similar to the `sudo` program. `Priv` accepts and executes a UNIX command, script, or binary as a different user ID. You need to set up a configuration file that determines who can run what commands in order for `priv` to work. The main difference between `priv` and `sudo` is that you do not have to enter a password to execute the command. `Priv` allows you to have asynchronous, on-demand command invocation through something like `cron`. It also provides a stable, repeatable execution environment. Two security features of `priv` are:

1. It is self-checking: Defensive Programming Style.
   - scrubs its environment (IFS, PATH, TERM)
   - is very critical about its config files' ownership and modes
   - uses absolute path invocation
   - has indirection protection (Program must reside in a hard coded path.)
   - binds statically (There is no dynamic link/load module.)
2. Access is controlled by the site computer security/sysadmin.

A good application for `priv` would be to allow users to reboot their machines yet not give them access to the root pass-

word. Or perhaps the users need to run privileged jobs in batch mode.

You can find `priv` at <ftp://ftp.home.net/pub/swelch/priv.tar>.

Using Internet Standards to Control the Cost of Spam

William D. Yang, Greater Columbus Free-Net

Summary by Allen Canning

The main focus of this talk was on how to limit spam using standards (RFCs). There are two main ways that spammers work: relaying and forging addresses. Relaying is not a required function of SMTP as per RFC 821, 822, and 1123. Forging does not comply with the RFCs; we are supposed to be able to determine who sent a message using the headers.

There are couple of ways that we can control spam. Stop relaying mail or at least restrict whom you allow to relay mail. After you have dealt with relaying, you can use sendmail to filter out invalid mail messages before they are written to your disks. Once sendmail is set up, it is time to educate your users. Having them use `procmail` or another MUA (Mail User Agent) to filter unwanted mail will help reduce the number of support calls that you have to respond to.

There are some caveats to filtering mail, namely, censorship. Any spam filtering should be done with the cooperation of your company's legal counsel.

Here is a list of references the speaker provided:

<http://www.sendmail.org>

<http://www.informatik.uni-kiel.de/~ca/email/check.html>

<http://info.internet.isi.edu/in-notes/rfc/files>

<http://spam.abuse.net/spam>

<http://www.cauce.org>

<http://www.gcfn.net/spam>

If you wish to receive sample code via autoresponder, send mail to either of the following email addresses:

gcfn.server.m4: <mailto:spam-m4-request@gcfn.net>

procmail rules: <mailto:spam-rules-request@gcfn.net>

Experiences Learned Securing a Web Server

David L. Kensiski, Deer Run Associates

Summary by Allen Canning

The goal of this talk was to describe the process of securing Cisco Connection Online (CCO). CCO is the conglomeration of servers that make up Cisco's Web presence. CCO gets about four million hits a day and is responsible for over one-third of Cisco's total business. It was very important to the company that it be secure.

First, the network had to be secured. This was done using filtering routers. The first router (gw1) provided a basic ACL (access control list) to prevent IP spoofing. The DMZ (De-Militarized Zone) between gw1 and Cisco's intranet was protected by another router (gw2). This router blocked all unnecessary services. Gw2 basically was the corporate firewall. Also on the DMZ was a router (gw3) protecting CCO. This router (gw3) blocked all services except the ones needed for CCO. There was one more router (gw4), but it was located inside the firewall. It protected Cisco's intranet from the EC (Electronic Commerce) machine. This router allowed communications to go only from the EC machine out to CCO, thus protecting the internal network.

The communication that took place between the CCO and the EC machine is called STA (Secure Transport Architecture). It was another topic at the conference. This is the only traffic that is allowed through gw2, gw3, and gw4.

The next step was to secure the host itself. All nonessential services configured in `/etc/inetd.conf` were turned off. No NFS or rpc was allowed. All administrative communication was done via `ssh`. To copy files to the system, there was a Kerberized `ftpd` setup that was wrapped to allow connections only from inside Cisco.

Now they had to set up a list of procedures. Most of the procedures applied to how system administration was performed. A strict root password policy was adopted. The root password was used only on the console and only in emergencies. All other privileged access was controlled by sudo. RCS was used to control all system files. There was also a homegrown application (`supercrc`), similar to `cops`, that was used to notify the admin staff when files had been modified.

One of the better procedures that was implemented was a Code Review. Any and all code that was to be placed on the system (applications, CGI, etc.) had to be sent in for review. There was a 48-hour turnaround on code submitted, and there was an escalation process. The escalation process not only raised the priority of the review, but notified the developer's manager. The hope was that only critical pieces of code would get escalated. All the code submitted was stored in a database that allowed for tracking and ownership. If some code was misbehaving, it was pretty simple to look up who owned it.

This project succeeded in securing the Web site. But the best part of the whole project was the development of a mechanism whereby any code on the system was guaranteed to be secure and could be tracked back to the owner.

The World Wide Web GhostTown

John N. Stewart, Digital Island Inc.

Summary by Allen Canning

This talk focused on two points: Web sites becoming outdated and Web sites facing digital destruction. Digital destruction occurs when a revision of a Web site is replaced or deleted and no copy is saved.

Several Web sites are outdated or just old. It's not just Web sites. WebTrends shows that 10% of people still use Netscape 2.x as their browser. Another large problem is that data are too extensive to be accurately stored in the current search engines. Searches may have to be limited to a date range. The main problem is old information. Here are a few examples:

- Live from Lilehammer (1994 Olympics)
- Blizzard of '96 (Boston)
- *Terminator 3* (this movie was never even released!)

Some of the questions raised in this talk are: "Are there going to be HTML collectors, similar to coin or stamp collectors?" "Will our children be able to do any research on our times?" "Are the Web sites of today being saved anywhere?" These are some very interesting questions that should be answered now, not later.

Dealing with Your First Break-in: Mistakes Made, Lessons Learned

Steve Remsing, Raytheon STX, Laboratory for High Energy Astrophysics, NASA/Goddard Space Flight Center

Summary by Allen Canning

This talk offered many excellent tips concerning securing your current systems and the appropriate steps to take if your systems ever suffer a break-in. Another point was the importance of working with your users and management to build policies and procedures to improve the security of your network.

Some of the configuration issues that help prevent system break-ins are selecting hostnames, performing routine system audits, patching your systems, and logging. Selecting "good" hostnames can be crucial. Never use system names that describe a system function (i.e., `nfsserv`). Do not name your systems sequentially; if `sales1` gets hacked, there is a pretty good chance that `sales2` will be attacked next.

In order to respond to a break-in, you need to first determine if you have been hacked. This can be done by searching through your log files, checking to see if any have been modified on disk. A system that starts to behave strangely is another indication you have been hacked.

A central logging server can be crucial to determining what happened to a system. It is relatively easy for hackers to modify any and all traces of themselves once they have compromised your system. With a central logging server, you get a nice audit trail.

Once you determine that you have been hacked, you need to plan your response. Are you going to go after the hacker in court? Should you leave the system in a compromised state to use as evidence? The first thing to do is make several backup copies of the system. Then start to gather all the information you can about the hackers. Where did they come from? How did they get in? What other systems were they successful in compromising?

Now that you have some evidence, you need to reprovision the compromised system. There are two ways this could be done: restoring from backup or reinstalling from CD-ROM. Remsing recommends reinstalling from CD-ROM. This eliminates the chances of the exploit getting reintroduced to the system.

After you have rebuilt the systems, it is time to evaluate your security policies and procedures. Take the opportunity to make management aware of the risks associated with a relaxed security policy. Explain to your users the correct security procedures, and the possible results of not following those procedures.

The speaker provided the following links:

<http://lheawww.gsfc.nasa.gov/~srr>

<http://www.cs.purdue.edu/coast/coast.html>

<http://www.cert.org/>

<http://nasirc.nasa.gov/>

<http://ciac.llnl.gov/>

<http://www.iss.net/xforce/>

Firewall-1 Address Translation Without Proxy ARP

Bill Canning and Edward Jones

Summary by Charles Gagno

The goal was to design a fault-tolerant firewall solution providing 99%+ uptime on the Internet connection and automatic failover for firewall boxes. That's exactly what Bill Canning and Edward Jones did using two Firewall-1 boxes running on Sun workstations. Their design includes an "access route," connected to the Internet and the external DMZ, two firewall boxes linking the external DMZ with the internal DMZ, and a "choke router" routing between the internal DMZ and the internal network.

Their design is based on a "Fake Net," a virtual network used for address translation preventing the use of Proxy ARP. Let's assume:

172.16.1.0	External DMZ
172.16.2.0	Internal DMZ
172.16.3.0	Fake Net
10.0.0.0	Internal Net

The design follows these simple rules:

- The access router routes everything from the Internal DMZ and the Fake Net to the firewall.
- The firewall routes everything form the Internal Net and the Fake Net to the choke router.
- The firewall's default route points to the access router.
- The firewall routes the Fake Net to the choke router.
- The choke router routes everything to the internal interface of the firewall.

When a packet comes in, the access router sends it to the Firewall box. The firewall does the address translation and sends the packet to the choke router. The

choke router then sends the packet to the real recipient.

The advantages are: Proxy ARP is absent, failover is based on routing only, new "address translated" hosts can be added without any special configuration, and the concept allows more "address translated" hosts than Proxy ARP-based solutions.

You can reach Bill Canning at <bill.canning@edwardjones.com>.

Oracle Database Management for Systems Administrators

Scottie Swenson

Summary by Charles Gagno

Oracle can be a scary thing for the normal system administrator. Because Oracle is in practice an OS on its own, Scottie made a presentation to explain to systems administrators what Oracle is and how it works.

He gave a good overview of the nomenclature used in the database world, describing what DB, RDB, RDBMS, tablespaces files, instances, and various internal elements are.

He gave a quick overview of the Oracle structure, explaining the role of every component from the SGA (System Global Area) and PGA (Program Global Area) all the way to the data block buffers needed for I/O access.

Let's look at the different Oracle files. It's important to mention that some of these files can be configured on raw devices:

- parameter
- data files
- control files
- Online Redo Logs
- Archived Redo Logs

The parameter file is usually named after the Oracle instance (`$ORACLE_HOME/dbs/initDBS.ora`). It's read at Oracle startup, and it controls the setup of SGA resources and Oracle server parameters. The control file contains the Data Dictionary and is required for database operation. The Online Redo Logs are chronological records of each transaction. A copy of the Online Redo Logs is stored in the Archived Redo Logs.

A lot of processes are involved in each Oracle instance. The SMON is the system monitor, and it performs automatic instance recovery, reclaims space used by temporary segments, and merges contiguous areas of free space in the datafiles. The PMON is the process monitor. It works with SMON and cleans up abnormally aborted connections. Finally, DBWR is the database writer, and LGWR is the log writer. A lot of user processes with associated PGAs are also involved with various tasks on an Oracle database.

Swenson's presentation is available online at <http://cellworks.washington.edu/pub/Presentations/199805_SANS_ITALK.pdf>.

Scottie can be reached at <swenson@u.washington.edu>.

How to Move to a New Building and Keep Your Users Satisfied

Marcel-Franck Simon

Summary by Charles Gagno

Moving can be a real puzzle for IT departments. A building move has to be seen as a discontinuity in the delivery of IT services, and it has to be planned that way.

The first task is assembling a team. The functions of the team are normally separated like this:

- physical network infrastructure
- logical network infrastructure
- servers and desktops
- PBX, phones, network, and voice connectivity
- program management
- interface to general contractor

Decisions will have to be made on the type of move (flash or gradual), how much infrastructure will be involved, and which technologies will be involved.

Make sure you stay involved with the general contractor who will be responsible for the construction, power, A/C, plumbing, and everything else in the new building. Even if the building is not new, some modifications requiring a general contractor will be needed in the new computer room.

Subcontractors will most probably be involved:

- network cabling installation vendor
- network hub/switch vendor
- PBX vendor and/or mover
- computer mover(s)

Before the move, it important to perform upgrades needed on the actual infrastructure because things won't change for a certain amount of time. It is also crucial to carefully plan how things will be laid out in the new location. It is recommended to write a detailed move schedule and communicate it to the users.

Right before the move, make sure the schedule is all set, and prepare a loading dock access priority. You have to make sure trucks carrying high-priority items (e.g., network switch or DNS server) will have priority over trucks carrying books. Also make sure you have proper insurance for all the equipment, and prepare replacement components.

On moving day, keep the working hours reasonable. After 12 hours of continuous work, workers are more prone to make fatal mistakes. Spend a lot of time looking for problems. Doing this allows you to stay out of the way and to prevent more important problems potentially fatal to a successful move. Make sure you keep management and the users informed of the progress. Get phone connectivity as early as possible in the

process; it will be helpful in case something happens.

The day after the move, come in early to make sure everything is okay, and provide a walk-up help desk for both IT and non-IT problems. Make a good follow-up on the move in general.

The week after, finish cleaning up, and keep communicating with the users.

The Architecture of the DNS Service for the Philips Intranet

Jim Reid, Origin TIS Intranet Services

Summary by Brian Kirouac

Jim Reid presented his personal experience of how Philips Intranet, specifically, the Origin group, went from an "internal DNS zoo" to a system that was redesigned with the intention of "doing it right this time."

The Philips network is large, really large, with roughly 150,000 hosts and multiple sites. Each site was extremely varied, with the number of hosts ranging from less than 10 to more than 10,000. They had no real control on the growth of their DNS. They had just about every flavor of DNS server, with an untold number of DNS administration tools.

The model they chose for their new DNS was a centrally managed DNS backbone. Every machine was to be identical. And all servers were to contain everything within the internal DNS. This whole design fit the goals of adaptability, efficiency, robustness, reliability, simplicity, scalability, ease of administration, and, most of all, low cost of ownership.

This first decision was what operating system to use. A BSD-based UNIX was the winner. Some of the reasons are that everything "just works," all public domain tools are already there, commercial support is available, it is the reference platform for DNS, and a nameserver-friendly virtual memory subsystem.

The next decision was BIND 4 or BIND 8. Bind 8 was just released and unknown, but BIND 8 was chosen for a basic reason: not wanting to have to upgrade the system in the future. This decision was never second-guessed, as BIND 8 worked fine from the beginning.

The last decision was which platform to use. The system administrators wanted to use Sun's, but there is no BSD for sparcs. Thus the PC was chosen. DNS is not a compute-intensive system, but it is heavily RAM dependent. A swapping DNS server is overly slow.

The rest of the talk went into the specifics of how the machines were deployed throughout the worldwide enterprise, along with the specific ways of dealing with the DNS datafiles.

SAGE news & features

A Trip to SANS

by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections, and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

It's been four years since I last attended a SANS conference; that one was in Washington, D.C. in 1994. The recent SANS conference held in Monterey certainly showed a "coming of age" compared to those I attended in the early 1990s. The conference has grown in many ways.

In 1994 there were about 300 attendees; this time there were 1,400. But more than just attendance has increased. The number of sessions is up dramatically. If my memory serves me correctly, the early SANS conferences consisted of a handful of tutorials and a track of invited talks. This SANS had more of each: tutorials ran several days before and after the technical conference, which sometimes had as many as three invited talks sessions running concurrently. That more or less guarantees "something for everyone." I spent a lot of the day wishing I could be in four places at the same time. One criticism in this regard: the preconference and conference schedules were hard to follow. Part of that was page layout, but once I actually got to the conference and figured out what was going on, I realized that it was also because different tracks were breaking at different times. Comments I overheard in the hallways made me believe I wasn't the only one who found this confusing. Perhaps there's a good reason for it (better for hotel catering?).

SANS seems to have grown its audience, too. My perception was that the attendees came from a broader geographic area. The early SANS conferences drew heavily from the immediate D.C. area. This SANS still had a strong showing from the East Coast, but also seemed to have a lot of international attendees, as well as the West Coast locals.

There were more people-networking opportunities: movies, international BOFs, more food, snacks, pizza night, Rob's popular quiz show, etc. There was no shortage of BOFs to choose from. Some were more social in nature, but many were on technical topics. The one I attended included a brainstorming session, led by Alan Paller, director of research at the SANS institute, on what tools the community needs to monitor machines. There was a lively interchange on requirements and thoughts on the correct architecture of the solution.

I enjoyed several sessions on a variety of subjects. I listened with particular interest to Wietse Venema's talk on "Turning Off Sendmail Forever." He turned off sendmail on his machines in late 1997. After the talk, I asked him how many alpha sites were running his code, and he said about 200. He has carefully architected VMailer, his sendmail replacement, to be fast (enough), secure, compatible, and flexible. He also wants to make it easy to deploy, so he's going to give it away. I wasn't wild about the table-driven rewriting in version 1, but if anyone can write a successful replacement for sendmail, Venema's got to be considered a seriously capable candidate.

I also attended sessions ranging from antispam solutions to virtual private network solutions. In general, the quality and range of topics were good. Most of the people that I talked with had the same impression.

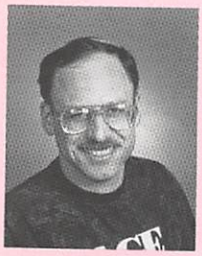
Partway through day one, I began to internalize a fundamental difference between SANS and LISA technical conferences. LISA is made up of refereed papers and invited talks; both are reviewed and selected by the conference committee. Conversely, SANS has just a handful of peer-reviewed talks, with the bulk of the sessions being invited talks. I'm not sure that fact alone is critical, but I do think it results in a different "flavor." On average, the kinds of tips and insights you'll get from a LISA presentation is a hands-on, vendor-independent, "from-the-trenches" slice at a solution to a problem; it's usually something the presenter has created and is willing to share. I noticed that many of the presentations I attended at SANS were commercial applications applied to problems. This isn't necessarily bad, but it's different.

So if you're a sysadmin, which conference do you choose? Of course, "that depends." In my mind, although both conferences are about the same general things, SANS and LISA are different, with different constituencies coming to each. If you're new to the field, there are lots of introductory sessions and opportunities to people-network at both conferences. If you're interested in new, freely available, solutions to "right now" problems, LISA's paper track is for you. If you just need to get your feet wet on a whole bunch of relevant topics, SANS will do the trick, too.

Overall, I really enjoyed this SANS conference. The staff and attendees were friendly; I ran into a lot of old friends and made a few new acquaintances. There were loads of things to do and learn about. Many of the sessions were informative and well-presented. And all this was just over an hour from my house, in one of my favorite venues in the world.

I got my money's worth.

The Case for Expansion



by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

When you consider the number of people out there who are really doing system administration work, the number we have as SAGE members is really pretty small. Membership is growing, but we have a long way to go to bring into the fold any significant portion of our potential constituency. Still, this is a good time for us to look at expanding into different "markets."

Our "community" is that of system administrators. It seems clear to most, if not all, of us that our definitions of the term "system administrator" vary widely, sometimes apparently mutually exclusively. We seem to include folks whose titles might vary from database administrator to network administrator, and then some. We have software tool builders and people who don't know how to program. There are consultants, contractors, and employees working in large or small teams or alone. We bundle it all together. But does everybody?

I think there are nonorganized communities out there now that could benefit greatly from getting involved with SAGE: database administrators, networkers, telephone switch engineers, and NT administrators, to name a few. We are just beginning the courtship dance with NT folks, mostly by trying to address concerns of UNIX admins who find themselves thrust into a combined environment. We are working on putting together a networking workshop/conference that will begin to address and attract that group or that side of "our" group. How about telephone and database folks?

Their needs approximately equate to our needs. In fact, a good many of "them" are "us" already – I have nearly always owned the telephone switch and generally get handed the database server onsite. Most of what each of these groups does is user support and care and feeding of complex automated service centers. All lead high-pressure, fun-packed, problem-filled work lives. All face similar vendor choice patterns, upgrades, patches, relocations, and problem-solving issues. Nearly always now the equipment each group supports is connected directly to that supported by the other groups. Them Is Us, except that an even larger percentage of Them is not involved in a SAGE-like organization than is true with the sysadmin community.

We need each other. The days of computing as we knew it are gone. A great percentage of current SAGE sysadmins were (and still are where they can be) operating in a "the computer is the subject matter" environment. That is changing, and fast. Computers are finally becoming only the tools applied to other subjects rather than ends unto themselves, and the shift in emphasis is going to be important to understand. Those who are "true" dba's or telephony engineers may understand this already because their fields have long been oriented to the "service provision" side more than "tool development." SAGE understands how to solve problems as a community and even understands something about what those problems are.

When I'm faced with a confusing problem on a platform I'm less familiar with, I go to a SAGE colleague who has relevant experience. What I'm suggesting here is the same thing: bring in more expertise in these related areas and share the benefits.

I don't want a dozen tracks at LISA or to see LISA split into lots of small conferences where we lose the synergy of our annual "gathering of the clan." I just want to recognize that there are people out there in very related fields with whom we share too much to ignore. It is time to consider how to include them in SAGE.

SAGE, the System Administrators Guild, is a Special Technical Group within USENIX. It is organized to advance the status of computer system administration as a profession, establish standards of professional excellence and recognize those who attain them, develop guidelines for improving the technical and managerial capabilities of members of the profession, and promote activities that advance the state of the art or the community.

To achieve its mission SAGE may:

Sponsor technical conferences and workshops;

Publish a newsletter, and/or professional short topics series;

Develop curriculum recommendations and support education endeavors;

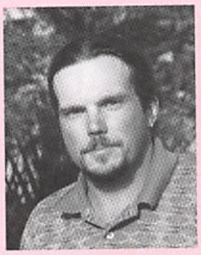
Develop a process for the certification of professional system administrators;

Recognize system administrators who are outstanding or are otherwise deserving of recognition for service to the professional community;

Speak for the concerns of members to the media and make public statements on issues related to system administration;

Promote and support the creation and activities of regional or local professional system administrators.

Effective Perl Programming: An Object of My Affection



by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

<joseph@5sigma.com>

I'm fond of object-oriented programming in Perl. Unlike a certain C-like object-oriented programming language you may have come to love (or hate), Perl has a very simple object-oriented programming framework. But simple as it may be, it is sufficient to cover all the essentials: classes, objects, methods, single and multiple inheritance, constructors, destructors, function overloading, and a few "frills" like tied variables and operator overloading.

This is too long a laundry list of features to cover all at once here, but in this column, I'll give you a whirlwind introduction to objects in Perl. If you're not too familiar with references, anonymous hashes, and the like, you may want to have a Perl reference of some kind at hand to help you puzzle through the syntax. Feel free to stop, type in the code, and play around with it as you go.

Creating Objects in Perl

A class in Perl is a Perl package. It's nothing fancier than that. To create an object that is a member of a class, you take a reference to some data and use the `bless` operator to "label" the data as belonging to that class. The data are often, but not necessarily, a hash. Hashes make convenient Perl objects because hashes are a good approximation of structured data types like structs or records:

```
package Person;        # make Person the default package
$obj = { 'first' => 'Joseph', 'last' => 'Hall' };
                       # $obj is a hash reference
bless $obj;            # bless $obj into Person
print "The person is $obj->{'first'} $obj->{'last'}\n";
                       # prints The person is Joseph Hall
```

Perl's object-typing mechanism is a runtime one. When creating an object, you must always explicitly bless it. The blessing operation generally occurs in a constructor, in combination with the process of creating the data in the object itself. A constructor is a subroutine, generally (but once again, not necessarily) called `new`, that is responsible for creating and blessing objects. Note that the constructor in the following example

Perl methods are just ordinary Perl subroutines written to certain conventions.

expects that the first argument passed into it will be the name of the class for which the constructor is written:

```
package Person;
sub new {                  # constructor for class Person
  my $class = shift;       # should be the string 'Person'
  my $self = { @_ };       # make hash ref out of rest of args
  bless $self, $class;     # blesses $self into Person and returns it
}
```

I've used the two-argument form of `bless`, where I explicitly specify the class that the object will be blessed into. This will come in handy when I get around to discussing inheritance. You *could* call this constructor like this:

```
# this is the "ordinary subroutine call" syntax -- see below
my $joe = Person::new('Person', 'first' => 'Joseph', 'last' =>
    'Hall');
```

I say you could, not *should*, because I haven't shown you the proper syntax for calling a constructor in Perl. That's a method call syntax, and that's coming up next.

Calling Methods in Perl

Perl methods are just ordinary Perl subroutines written to certain conventions. There are two types of methods: class methods and object methods.

A class method can be called with one of two different types of "method call syntax." The first is the so-called "indirect object" syntax, where the name of the method is followed by the class name, then the remaining arguments to the method. No comma follows the class name. Here's how we could call the `Person` constructor from above using indirect object syntax:

```
$joe = new Person 'first' => 'Joseph', 'last' => 'Hall';
```

Or, if you prefer:

```
$joe = new Person('first' => 'Joseph', 'last' => 'Hall');
```

Perl automatically translates this indirect object method call into the ordinary subroutine call syntax shown above – it calls the subroutine `new` in the package `Person` and prepends the string `'Person'` to the argument list. The second type of method call syntax is the "arrow syntax":

```
$joe = Person->new('first' => 'Joseph', 'last' => 'Hall');
```

This is equivalent to the indirect object form and is translated into an ordinary subroutine call in the same manner.

Object methods are similar to class methods, except that they take an object instead of a class name. I'll first define an object method for `person`:

```
package Person;        # default package is Person
sub first {
  my $self = shift;    # first argument is object
  $self->{'first'};    # return first name from a Person object
}
```

Assuming that `$joe` is still defined from one of the constructor calls above, we can invoke the method `first` with arrow syntax:

```
print "Joseph's first name is: ", $joe->first(), "\n";
```

Now, instead of supplying the class name as the first argument to the method, Perl supplies an object (in this case, `$joe`). And how does Perl know what class the method `first` is in? Simple – `$joe` has been blessed into the class `Person`, so Perl looks there.

Although it's possible to do so, I don't recommend using indirect object syntax for object method calls. Stick to the arrow syntax for object methods – the indirect object syntax has many potential pitfalls. You've been warned!

Inheritance in Perl

Method call syntax doesn't exist just for its eye appeal. If you call a nonexistent method, Perl will search an inheritance tree for other classes that might have a method by that name. For each class, the special array `@ISA` (pronounced "is a") contains a list of class names to search for inherited methods. Another way of looking at this is that for each class, `@ISA` is a list of that class's parent classes.

For example, assuming I'm still in the same file with the `Person` constructor above, suppose I have

```
package Programmer;
@ISA = qw(Person);
$john = new Programmer('first' => 'John', 'last' => 'Doe',
  'language' => 'Perl');
```

Now, even if this is all I have for the class `Programmer` so far, it will, amazingly enough, work. Because there is no subroutine `Programmer::new`, Perl will search the `Programmer` inheritance tree for other classes that define `new`, and Perl will come up with `Person::new`, and call it like this:

```
$john = Person::new('Programmer', 'first' => 'John',
  'last', => 'Doe', 'language' => 'Perl');
```

This is very important: the first argument Perl passes to `Person::new` is now `'Programmer'`, not `'Person'`, because `new` was called for class `Programmer`. The constructor I wrote for `Person` is inheritable because I used the two-argument form of `bless`. If I had used the one-argument form, `$john` would have been blessed into `Person`, which would be wrong. I can even make this "generic, inheritable constructor" a little more succinct (this is a pretty handy snippet of code, actually):

```
package Person;
sub new {          # the generic, inheritable constructor
  my $class = shift;
  bless { @_ }, $class; # combine anon hash and blessing
}
```

Thoughts on Object-Oriented Perl

There are many reasons why object-oriented programming in Perl is a much less complicated affair than it is in a language like C++. For one, Perl has built-in memory management. C++ constructors often devote a lot of code to memory allocation issues. Such code is extremely rare in Perl.

Another reason is that Perl does not support data inheritance. In Perl, only methods are inherited. This may seem curious at first, but it makes sense because Perl lacks rigid structured types like C's structs. Also along those lines, Perl lacks extensive "access control" features like the `public`, `private`, and `protected` members of C++ classes. Not having to make, and later modify, decisions about member access definitely helps simplify the process of prototyping object-oriented applications and modules in Perl.

In my next column, I'll discuss some ways in which you can implement some of the object-oriented programming features that Perl is apparently lacking – for example, private class data. I'll also show you a real-world example of subclassing and using an existing object-oriented module from the CPAN (Comprehensive Perl Archive Network).

A Note

Some of the material in this column is loosely derived from course materials that Randal Schwartz and I have written. As always, I'd like to acknowledge and thank Randal for his help and inspiration.

Toolman: Sorting and Archiving Email

In this article, I'll discuss a methodology for sorting email into mailboxes based on year and month, which can then be compressed for archival purposes. In addition, I'll cover retrieval techniques, and I'll survey some related tools.

by Daniel E. Singer

Dan has been doing a mix of programming and system administration since 1983. He is currently a system administrator in the Duke University Department of Computer Science in Durham, North Carolina, USA.

<des@cs.duke.edu>

Tools:	*sortmail, decomposemail, recomposemail*
Abstract:	*sort email messages by date/time into monthly mailboxes*
Platforms:	*most UNIX*
Language:	*Bourne shell*
Author:	*Daniel E. Singer* <des@cs.duke.edu>
Availability:	<http://www.cs.duke.edu/~des/toolman.html> <ftp://ftp.cs.duke.edu/pub/des/scripts>

Hoarding

If you're like me, you're a pack rat with your email: you stow it away somewhere, but never like to get rid of it or take it offline. After all, you never know when you're going to need to `grep` through it to find some vital instructions, reconstruct a conversation, or verify that you or someone said something 27 months ago. All this old email takes up a lot of disk space. And some of us live within quotas. (I'm currently struggling to stay within a 100MB quota, more on principle than necessity.)

So what's an email hoarder to do? Some people use any of various mail filters to automatically sort incoming email into mailboxes (sometimes known as folders) and even discard certain messages as they come in (can you say "spam"?). Examples are `procmail` [1, 2] and the bundled filtering features of `elm` [3]. But I'm kind of old-fashioned and distrustful of these filters: I like to decide on a case-by-case basis which messages to put where, and how long they should hang around in my *inbox*, saving or deleting them as seems appropriate. What tends to happen is messages pile up in my *inbox*, and periodically I'll go through and save some old messages to mailboxes and purge out others. As I do this, messages get saved out of chronological order – sometimes *very* out of order. This may or may not resemble your email processing practices.

To complete this picture, let me add that I save messages to mailboxes using filenames based on the username of the sender or the name of a company, product, or concept, along with certain upper- and lowercase conventions. (Occasionally, I'll even save a message to more than one mailbox because the concepts of links and cross-posting are not available in this context.)

Sorting and Chunking

What I want is a way to save my email, archive it in manageable chunks, compress it, and still keep it useful. (Yes, I want to have my cake and eat it!) I could just periodically move mailboxes to an archive directory, add sequence numbers to the filenames, and compress them; but each such archive would not necessarily be sequential over some period, and searching would be more difficult than it could be. I want to be able to search through email by time periods as well as by some person or topic. Also, some mailboxes tend to get very large and unwieldy (that is, slow), so splitting them into chunks should also be a performance gain.

The methodology I've come up with for this is to disassemble mailboxes into their component messages, sequence them by date/time, and then reassemble them into mailboxes by year/month, optionally storing these into monthly subdirectories. For instance, I have a mailbox named "USENIX," and since it tends to collect a lot of messages, I occasionally want to *chunk* it (not *chuck* it!). I can do this by going to my mail directory, and running the `sortmail` script.

```
% cd ~/mail
% sortmail -mc USENIX
```

This will create (or append to) mailboxes with names like "USENIX.9805" for May of 1998, "USENIX.9806" for June of 1998, and so on. Each such mailbox will hold the messages for that month of that year only, sorted meticulously by date and time. Alternatively, we could have used `-M` instead of `-m`, telling `sortmail` to instead deposit

the sorted email into monthly subdirectories, yielding mailboxes such as "9805/USENIX," "9806/USENIX," etc. In either case, `sortmail` will append to the monthly files if they already exist. And if any such monthly mailboxes are already compressed (via `compress` or `gzip`), `sortmail` will first decompress them, then add the new messages, and then recompress them. If the `-R` (recurse) flag is used, any appended mailboxes will also be resorted. The `-c` flag tells `sortmail` to move any messages for the current month back to the mailbox of the original name, in this case "USENIX."

I tend to prefer the YYMM/mbox scheme over the mbox.YYMM one, because I currently have around 500 mailboxes in my mail directory, and the latter scheme would add too much additional clutter.

Another tool similar to `sortmail` is similarly named `mailsort`. It is written in Perl by Andras Salamon (<http://www.dns.net/andras/>). `mailsort` can also reverse sort and is styled after the UNIX filter model much more so than `sortmail`. It is fast and robust, though it lacks the monthly chunking features of `sortmail`. You can pick up `mailsort` at your fave CPAN[4] site under <.../scripts/mailstuff/mailsort.tar.gz>.

Safeguarding

To safeguard your precious data (and mine), `sortmail`, by default, will also: (1) create a subdirectory into which it backs up any mailboxes that it is going to change and (2) create an additional subdirectory into which it copies mailboxes to be sorted and in which it does all of its work. So if you're a little nervous about letting this stuff loose on your mailboxes, everything is covered. (Of course, you can make additional copies or copy the mailboxes to another directory and do it there until you get the feel of it.) After running `sortmail`, you can verify that things are OK, and then remove the backup copies. Then you can compress any of the older mailboxes if desired.

Supporting Cast

The `sortmail` script is a higher level interface to two scripts that do a lot of the work: `decomposemail` and `recomposemail`. Their names are indicative of their functions: the first breaks up a mailbox into files, each containing an individual message; the second reassembles the messages in sorted order. They each can be used standalone, though `sortmail` saves many manual steps and does add additional functionality such as making backups, working in a subdirectory, appending to existent files, and recursing.

Searching

Now, let's say you've been using `sortmail`, and you have subdirectories such as "9601," "9602," ..., "9807." Furthermore, you have already compressed all the mailboxes in the subdirectories for the months in 1996 and 1997. Now you want to find that cornbread recipe that your mom emailed you a year or two ago (and you don't feel like calling). Well, you don't want to go and uncompress all those files, and you probably don't want to type a bunch of awkward commands like:

```
% gzcat 9701/mom | grep -i cornbread
```

A tool that you can use for this sort of situation is `grepz`. It will uncompress on the fly (without modifying your files) and can even recurse through a directory hierarchy if given half the chance. So the line

```
% grepz -i cornbread 9[67]??/mom
```

would do the trick. In the event that you didn't know who sent you that recipe or when, a bigger hammer would be

```
% grepz -i cornbread .
```

What I want is a way to save my email, archive it in manageable chunks, compress it, and still keep it useful.

More Tools:	*grepz; rotatemail; check*
Abstract:	*search for patterns; rotate files monthly; maintain index files*
Platforms:	*most UNIX*
Language:	*Bourne shell*
Author:	*Daniel E. Singer* <des@cs.duke.edu>
Availability:	<ftp://ftp.cs.duke.edu/pub/des/scripts>

If you've got too many mailboxes and other files and subdirectories under your mail directory, another problem can be just keeping track of what's what.

This would search through all files and subdirectories recursively. `grepz` will also handle noncompressed files properly.

A similar search tool that can handle compressed files is a Bourne shell script named `zgrep` that comes with the `gzip`[5] utility archive. Another very handy search tool, written in Perl by Jeffrey Haemer and Jeffrey Copeland, is named `mgrep`[6] and is designed specifically for searching mailboxes. It returns entire messages that are matched, instead of just matched lines. These could be combined with `find` and `xargs` to approximate the recursive behavior of `grepz`.

```
% find . -name occult\* -print | xargs mgrep -i voodoo | less
```

A more generalized approach for dealing with compressed files is the `zloop` shell script by Jerry Peek. You can tell it to run the command of your choice on a group of compressed files. `zloop` is discussed in the book *UNIX Power Tools*[7].

```
% zloop 'mygrep -3d "on the road"' outbox.*.gz
```

Rotating

Another script that operates in this scheme of things is called `rotatemail`. I use it at the start of each month via UNIX's `cron` utility to automatically rename my "outbox" file congruent with `sortmail`'s monthly naming scheme. Sorting isn't necessary here since outboxes tend to be sorted already. A `crontab` entry like

```
0 0 1 * * rotatemail /home/you/mail/outbox 2>&1
```

will rename your outbox to "outbox.9807," assuming that July 1998 just ended. It will then create a new, empty outbox file with appropriate permissions. If you prefer the monthly subdirectory scheme, yielding a filename like "9807/outbox," then just add the `-M` flag. Of course, this could be used on files other than just your outbox.

Tracking

If you've got too many mailboxes and other files and subdirectories under your mail directory, another problem can be just keeping track of what's what. I've recently started using `check`[8] to create and maintain an INDEX file in my mail directory. This helps me to have a short description of each mailbox, to group them into categories, and to isolate duplicates that can be combined and junk that can be deleted. You might find this useful as an additional means of riding herd on your mailboxes. Then again, there's always that memory enhancement course you've been meaning to take!

Ending

A lot of territory has been covered here. My hope is that you can mix and match these tools and techniques to suit your taste. You might even want to add a few of your own design.

If you find that any of my tools don't work properly on your UNIX platform, drop me a line, and I'll pound on them for you. Just ask Bruce Foster at Northwestern University (<http://charlotte.acns.nwu.edu/bef/>). I recently fixed `seepath`[9] to work in his HP-UX/DFS/Posix-shell[10] environment!

I have a few other scripts that deal with mailbox manipulations. I'll leave them at the FTP location in case you're interested. As usual, please let me know if you have any comments or suggestions.

Notes

[1] `procmail` is written by Stephen R. van den Berg (<berg@pool.informatik.rwth-aachen.de>) at RWTH-Aachen, Germany, <ftp://ftp.informatik.rwth-aachen.de/pub/packages/procmail/procmail.tar.gz>.

[2] An interesting article about `procmail` and email filtering in general is Jeffrey Copeland and Jeffrey S. Haemer. Work: Not Looking Through Our Mail. *SunExpert Magazine*, May 1998, pp. 72-75. <http://www.alumni.caltech.edu/~copeland/work.html>.

[3] `elm` is maintained by the Elm Development Group, <http://www.myxa.com/elm.html>.

[4] Comprehensive Perl Archive Network. See <http://www.perl.org/> for the site nearest you.

[5] `gzip` is maintained by the Free Software Foundation. See <http://www.fsf.org/order/ftp.html> for the site nearest you.

[6] Jeffrey Copeland and Jeffrey S. Haemer. Work: Looking Through Our Mail. *SunExpert Magazine*, March 1998, pp. 80–84. <http://www.alumni.caltech.edu/~copeland/work.html>.

[7] Jerry Peek, Tim O'Reilly, and Mike Loukides. *UNIX Power Tools*, 2nd ed. Sebastopol, CA: O'Reilly & Associates, 1997. <ftp://ftp.ora.com/pub/examples/power_tools/unix/split/zloop>.

[8] Daniel E. Singer. ToolMan's Approach to Documenting UNIX Directories. *;login:*, June 1997, pp. 45–48.

[9] Daniel E. Singer. ToolMan: Upcoming Tools; Analyzing Paths. *;login:*, February 1998, pp.40–43.

[10] The Bourne shell drops implicit null arguments when parsing a string into positional parameters. The Posix-compliant shell on HP-UX (and possibly other platforms) does not, and this was causing `seepath` to choke.

Got a tool that's useful, unique, way cool? Please send a description to <Toolman@usenix.org>.

Call for Nominees for Election to the SAGE Executive Committee

SAGE is accepting nominations for members of the SAGE STG Executive Committee until October 15 at noon, PST. Anyone interested in running for the SAGE board should send his or her name and telephone number and a brief statement to the nominating committee via email at: <sage-nomcom@usenix.org>. You can also send U.S. Mail to the SAGE Nominating Committee care of:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

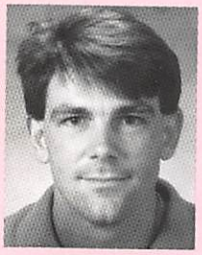
The nominating committee will gather the candidates' names and contact each of them before the election takes place.

In this election, directors will be chosen for 2 year terms (beginning in early 1999). This is the first election held under the new SAGE policies; all 7 seats will be elected (in the past, elections were yearly, with only half of the seats up in any one year). The SAGE Executive Committee chooses its own officers after each election, so all nominees run "at large".

At the LISA Conference, to be held December 6-11, 1998 in Boston, MA, there will be a candidates' forum to enable candidates to introduce themselves and talk about the issues. Candidates unable to attend the LISA conference will be able to submit a position paper to this forum. All candidates will be expected to respond for publication to a set of questions presented by the Nominating Committee. There will, in addition, be an on-line forum (most likely an archived mailing list) to enable SAGE members to pose questions to the nominees.

The new Executive Committee will take office in early 1999. Current estimates indicate that the new board will have at least 2 face-to-face meetings a year, and other meetings via teleconference.

If you have questions about the nominating process, or what Executive Committee membership entails, please send mail to <sage-nomcom@usenix.org> or contact a current member of the SAGE Executive Committee (see <http://www.usenix.org/sage/people/Current-Board.html>).



by Phil Cox

Phil is a member of the Computer Incident Advisory Capability (CIAC) for the Department of Energy. He also consults and writes on issues bridging the gap between UNIX and Windows NT.

<pcc@ntsinc.com>

TCPDUMP: The Spanner Wrench of Network Monitoring

Note: A spanner wrench is a tool that can do almost anything, if *you have enough creativity. I used them while in the Navy.*

I have been busy dealing with Denial-of-Service attacks during the last couple of months. A colleague of mine was tasked with writing a detector for the teardrop2 attack so people could determine if someone was running it against them. He did a great job, but then had to travel. I was tasked with "cleaning" it up and making it pretty. During the process, it occurred to me that I could do this same detection with a single tcpdump command line. This article is about how to leverage this flexible command to detect suspect packets on your network.

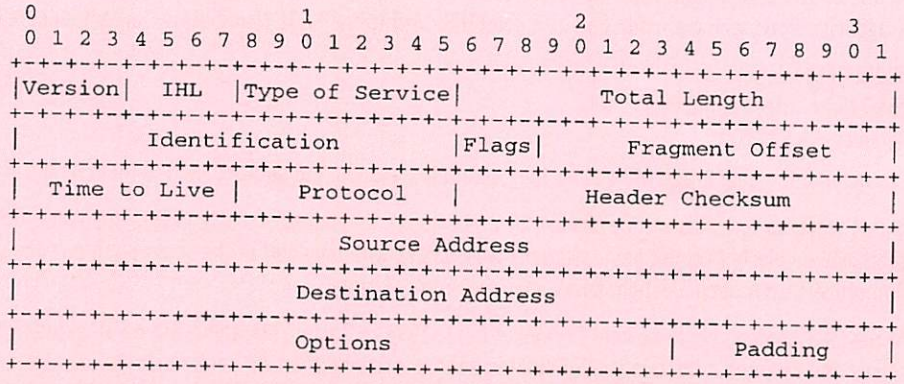
One assumption I have made is that the reader is familiar with IP/TCP/UDP packet structures, options, and flags. If this is not the case, this article might have some confusing spots in it. I *highly* recommend *TCP/IP Illustrated*, Vol 1, by W. Richard Stevens for those who need a more detailed reference and explanation.

Packet Formats

For reference, let's examine the formats of IP, TCP, and UDP packets[1]. See their respective RFCs for really gory details.

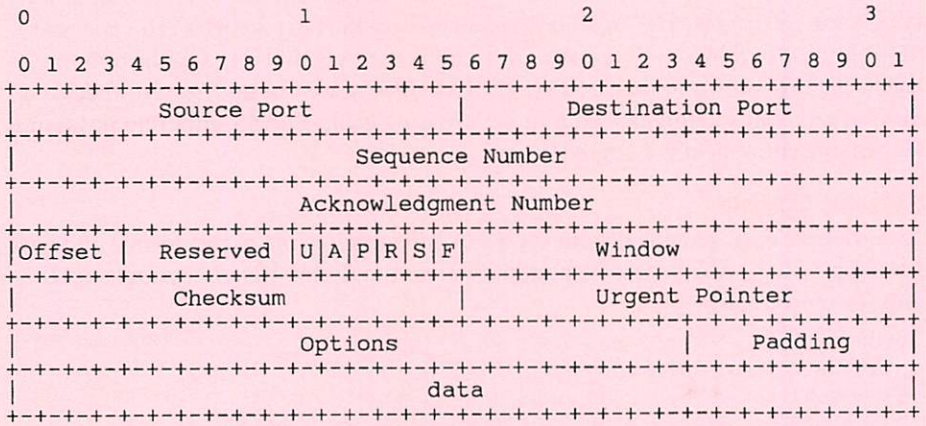
IP (RFC 791)

Bit



TCP (RFC 793)

Bit



UDP (RFC 768)

Bit

Tcpdump Specifics

The default output for `tcpdump` is very sparse. In order to get the amount of data in the format we want, we use the `-x` and `-s` arguments. The `-x` output option displays the data in HEX using 2-byte chunks, so, for example, the first 6 bytes would be represented by three "chunks" like this: 0000 0000 0000. The `-s` option is the snaplen, which is the amount of data to capture from the packet. This value must be large enough to encompass all the fields we are looking for. `tcpdump` allows you to selectively look at data at the bit or byte level for a given protocol. The format for that selection is `proto[expr:size]` [2], where `proto` can be one of the following: `fddi`, `ip`, `arp`, `rarp`, `tcp`, `udp`, or `icmp`, indicating the protocol layer for the index operation. The byte offset, relative to the indicated protocol layer, is given by `expr`. `size`, is optional, and indicates the number of bytes in the field of interest; it can be one, two, or four and defaults to one. Heads up on the value for the byte offset, `expr`, as `tcpdump` starts counting at 0, like C, so the third byte would be represented by the number 2. That's how to use `tcpdump` to get output. Now let's look at a key to decipher the output.

The default output for `tcpdump` is very sparse. In order to get the amount of data in the format we want, we use the `-x` and `-s` arguments.

Once we have data, generated from `tcpdump`, and we know the packet format and have the mapping key, we can look for patterns we want to detect.

Mapping Packets to Hex Output from Tcpdump

Let's start with the IP portion of the packet header. The standard IP header has a length of 20 bytes[3] and thus will be represented by the first ten chunks of HEX output we will see from `tcpdump`. TCP or UDP headers take up the next 20 bytes (10 `tcpdump` HEX-chunks) or 8 bytes (4 `tcpdump` HEX-chunks), respectively[4]. On the opposite page is the breakdown of the `tcpdump` HEX-chunks, so we can understand what each HEX bit represents. (Note that all number ordering is Big-Endian, and if no position is defined, the entire field is assumed.)

A Specific Example

Once we have data, generated from `tcpdump`, and we know the packet format and have the mapping key, we can look for patterns we want to detect. Here is a sample of HEX output from `tcpdump`:

```
4500 00b2 4ea6 2000 8006 ee3f c0a8 4803
c0a8 4804 044c 008b 00e5 c3a2 43c1 cc20
5018 217f f6f9 0000 0000 0086 ff53 4d42
3200 0000 0018 0380 0000 0000 0000 0000
0000 0000 0028 feca 0048 42f1 0f42 0000
```

Let's break this output down into its component parts, in this case, IP and TCP headers. As we know from the packet format above, IP is first ten chunks: 4500 00b2 4ea6 2000 8006 ee3f c0a8 4803 c0a8 4804

Now we examine the individual chunk and break it into its packet fields, using the mapping key. Continuing on our example, we begin mapping to the fields, starting with the first of the ten packet header HEX chunks.

4500

IP version	: 4 (Should always be 4, unless you run IPV6)
Initial header length	: 5 (always 5, unless Options are present)
Type of Service	: 0

00b2

Total IP packet length	: 178 bytes (0xb2 = 178)

4ea6

Identification	: 20134

2000 (hex) == 0010 0000 0000 0000 (binary)

Flags	: More Fragments (bit 3) is set to 1
Fragment offset	: 0 (means it's the first fragment)

8006

Time To Live	: 128
Protocol	: 6 (TCP=6, UDP=11)

eef3

header checksum	: 61171

c0a8 4803 = c0 a8 48 03

source address	: 192.168.72.3

c0a8 4803 = c0 a8 48 04

destination address	: 192.168.72.4

Because there are no IP options, the TCP packet starts next. As described previously, TCP, with no options, will occupy the next ten HEX chunks. From our example output, this is:

```
044c 008b 00e5 c3a2 43c1 cc20 5018 217f f6f9 0000
```

Mapping Key

Chunk Number	Chunk Position	Packet Field
IP Packet		
1	1	IP version number
	2	Initial header length, number of 32-bit WORDS
	3-4	Type of service
2		Total IP packet length
3		Identification
4	Special case: You have to convert to binary	
	1	first 3 bits are Flags Bit 1: reserved Bit 2: Don't Fragment bit Bit 3: More Fragments bit
	1	fourth bit is part of Fragment offset
	2-4	Fragment offset (measured in units of 8 octets, 64 bits)
5	1-2	Time To Live
	3-4	Protocol
6		header checksum
7,8		source address
9,10		destination address
11		IF IP options are present, they would start here and end on 32-bit boundaries (padded if needed). ** *Otherwise* (and most likely the case) this is the start of the IP data
UDP Packet		
11,12		Source Port
13,14		Destination Port
15		Length
16		Checksum
17 - n		UDP Data
TCP Packet		
11		Source Port
12		Destination Port
13,14		Sequence Number
15,16		Acknowledgment Number
17	1	: Offset
	2	: reserved
	3	: Reserved and Bit flags Bit 1,2: Reserved Bit 3 : Urgent Bit 4 : Ack
	4	Bit 1 : Push Bit 1 : Rst Bit 1 : Syn Bit 1 : Fin
18		Window
19		Checksum
20		Urgent Pointer
21 – n		TCP Data, unless TCP options

Armed with the mapping key and an understanding of how to use it, you should be able to identify any/all IP/TCP/UDP packet field values from the HEX output of a `tcpdump` session.

Just as with the IP portion, we break the TCP header chunks into fields, using the same mapping key. Beginning with the first of the ten TCP header chunks, we have:

044c
Source Port : 1100

008b
Destination Port : 139

00e5 c3a2
Sequence Number : 15057826

43c1 cc20
Ack Number : 1136774176

5018 (hex) == 0101 000000 0 1 1 0 0 0 (binary)
Offset : 5 (0101)
Reserved : (000000)
Urgent bit : 0
Ack bit : 1
Push Bit : 1
Rst bit : 0
Syn bit : 0
Fin bit : 0

217f
Window : 8575

f6f9
Checksum : 63225

0000
Urgent Pointer : 0

Because there are no TCP options, the TCP data would be the following chunks. Now that you have the hang of it, you could go as far as you desire in decoding the encapsulation of the packet, but I'll stop here.

Some Useful Applications

Armed with the mapping key and an understanding of how to use it, you should be able to identify any/all IP/TCP/UDP packet field values from the HEX output of a `tcpdump` session. What next? Well, let's say we want to capture all fragmented UDP packets (which is the basis for the teardrop exploit), but not the first fragment. To capture this type of packet, we could use the following tcpdump command line:

```
tcpdump -s 500 -x udp and '(ip[6]|0xdf = 255 and ip[6:2]&0x1fff > 0)'
```

`tcpdump -s 500` grabs 500 bytes of the packet, which is more than enough to get the UDP headers. Next we ask for a peek at the HEX contents of the packet, as described earlier, with `-x`. Further, we specify UDP packets and apply some selective logic. Recall from the "Tcpdump Specifics" section, that `tcpdump` allows you to selectively look at data at the bit or byte level for a given protocol with the format of that selection being `proto[expr:size]`. So, ip[6] will correspond to the seventh byte in the IP packet. Referring to our IP packet format we see that the 8 bits in this byte correspond to the 3 flag bits and the first 5 bits of the fragment offset. From our mapping key, we know that the third flag bit is the more fragments bit. To mask the byte to select only packets with the More Fragments bit set, you can use a bit mask of 11011111 OR'd with the value in your byte. If it's = 255, the bit is set. Similarly, the fragment offset is represented in the

fourth bit of the seventh byte of the ip header. It is a 13-bit field (5 bits remaining after the flag field, plus the next 8 bits). To detect if any of those 13 bits is set, we can AND the binary string 0001111111111111 (0x1fff) with ip[6:2] (i.e., 2 bytes' worth of the seventh byte of the IP packet). All told, a quick and dirty command line to look for miscreant, fragmented UDP packets.

Conclusion

With all the Denial-of-Service and other type of attacks that are roaming around the Internet, having a tool that is flexible enough to capture almost any IP data you can think of is of extreme value. Everyone needs the ability to detect attacks. Most systems I know of have a port of `tcpdump`, but not all have an Intrusion Detection System (IDS). `tcpdump` doesn't come close to replacing a proper IDS, but in a pinch, with an understanding of the attack, a creative `tcpdump` command line will go a long way.

Notes

[1] These diagrams were taken from the respective RFCs.

[2] See the `tcpdump` man page for more information.

[3] Currently, IP options are almost never used. If the Initial Header Length is not 5, then options are in use.

[4] This is only for the headers and considers that no TCP options are present.

A Day in the Life of a System Administrator

Mark your calendars now for September 15, 1998!

We all know that system administrators work hard and long hours. But what do they really do?

Contribute your day. It's simple; it's easy; it'll be fun. Check for instructions on how to contribute a diary of your work on September 15: see <http://www.usenix.org/sage/day/>.

SAGE will develop a typical day profile(s) and make this self-portrait of the sysadmin professional available to you. Maybe it'll help your boss understand what you really do.

by John Sellens

John Sellens is Director, Network Engineering, at UUNET Canada Inc. in Toronto, after many years as a system administrator. He is also proud to be husband to one and father to two.

<jsellens@uunet.ca>

On Reliability – Restores and Recovery

This time around I'd like to discuss some aspects of disasters, how to avoid becoming too much of a victim, and how to put things back together should your avoidance measures prove to be inadequate. I'm going to review things primarily from a system administration standpoint (this is the SAGE section of *;login:* after all), but it's important to remember that disaster recovery and avoidance is a far larger topic.

Computing professionals like us typically look at disaster recovery planning (DRP) primarily from a computing systems point of view, which is only natural when you consider which budget pool we're paid out of. Let's look at the big picture and hope that will help put the system administration issues into perspective.

What kinds of disasters might befall a company?

- the obvious ones: hurricane, flood, earthquake, explosion, etc.
- a tragic plane crash while all key staff are flying to a much-deserved off-site "retreat"
- primary product found to cause cancer in every living mammal except laboratory mice
- armed insurrection
- complete breakdown of municipal transit systems, preventing staff from getting to the office
- massive chemical spill and fire with toxic fumes at the company's plant, resulting in mass evacuations, health and environmental concerns, and virtually unlimited personal liability for the senior management and directors
- disgruntled key employees who start a competing company and lure away every employee with a non-zero IQ
- loss of the keys to accounts receivable filing cabinets, leading to billion-dollar write-offs

There's a lot more to DRP than making sure that the computers are running and the printers are printing. That said, let's concentrate on DRP for computing systems.

As I've tried to get across in the previous articles in this series, the key to appropriate levels of reliability is the balancing of the exposure to and costs of risks with the costs of avoiding those exposures. What's the worst that can happen (typically)? The company goes out of business, everyone is unemployed and without a pension, and the boss goes to jail. There's a story, which is probably apocryphal, of a mid-level executive who was charged with disaster recovery planning for his organization. His DRP? Keep an up-to-date copy of his resume at home. Most of us, however, would probably prefer to have at least something in place to provide some protection and recoverability.

Let's try and divide the problem space into three major areas:

1. major physical damage to computing hardware or communications infrastructure
2. utility (power, HVAC, telecommunications) failures
3. physical inaccessibility due to weather, structural damage to the building, civil unrest, or evacuation (due to chemical spill, fire, etc.)

The key to dealing with such problems is planning and documentation (on paper, both on- and offsite). Leaving your DRP until disaster strikes only increases its severity.

Physical Damage

If your entire computing infrastructure consists of a single clone PC, a modem, and a cheap printer, it's probably not worthwhile worrying too much about protecting your equipment from loss or damage. If something gets damaged, just go to any of the consumer electronics stores in your area and get a replacement off the shelf. If, however, your equipment is not typically available at the mall on a Saturday afternoon, you probably want to consider how to limit your potential damage and how to get access to replacement equipment in a timely fashion.

Physical damage to your computing and communications equipment can happen in a number of ways. Two of the most obvious are fire and water, but there are a number of other possibilities that you might consider protecting against.

Fire and Smoke

The best protection against fire and smoke is a safe, fire code-compliant building and an appropriate fire detection and suppression system. In past years, Halon was widely used as a fire suppression agent in computer rooms, but it was not environmentally friendly. Fire suppression systems are currently available based on carbon dioxide and other chemicals, but water-based sprinkler systems are still the most common suppression method. You might also consider the use of an emergency power-off system, to power down your systems in the event of an alarm. Among other things, this will help avoid damage to your equipment from smoke and residue being drawn into the chassis through the cooling fans.

Water

There are a few ways for water to attack your equipment. Plumbing failure is probably the most common, but you may also wish to worry about water damage from fire suppression systems or flooding. You should consider two primary attacks from water: falling down from above and seeping up from below.

From above, there are burstable pipes (both on your premises and feeding the bathtub or dishwasher in the unit above) and fire hoses. I've seen some installations with drainage trays mounted under the pipes in the computer room, draining off to the side of the room. And if you have control (or knowledge) of whatever it is in the rooms above you, you might want to worry about whatever plumbing there is up there. Otherwise, your best protection is to keep your equipment in racks or cabinets with a roof over them (and make sure that the ventilation fan outlet isn't in the middle of the top of the cabinet).

From below, consider a raised floor, with in-floor drains (including backflow prevention valves), pedestals, or some other device to keep your electrical connections off a potentially wet floor and an alarm system to warn you when it gets wet. And if your computer room is below grade, you may wish to reconsider its location. The farther you are above the water table, the safer you are.

Earthquake, Tornado

Two approaches to these problems are building integrity and equipment safety. If you are located in an area that is at risk for earthquakes or tornados, consider how your building would be affected if either hits. What parts of the building are most likely to be damaged – large plate glass windows, overhangs, trailer parks? Try to locate your equip-

You might also consider the use of an emergency power-off system, to power down your systems in the event of an alarm.

It's important to remember that, in a disaster, you won't be worried only about your central computing systems. . . . This isn't just a system administration issue – it's a facilities-wide issue.

ment as far away from these as possible. Consider bolting your equipment down in some appropriate fashion, and don't forget to fasten your rolling equipment racks down, too. There's no sense having your equipment bounce across the room or fall out the window every time there's a tremor.

Vandalism

Depending on your industry and location, you may wish to consider what vandalism, looting, revenge, or a disgruntled employee might do to your equipment. Is your computer room unlocked? Do you have big glass display windows to impress random strangers? Do you store a selection of fire axes in and near your computer room?

Alternatively, are you careful to collect keys and change security codes when an employee leaves (or is pushed)? Can employees enter on their own, or are two people required to act together to gain access to the computer room? Do you have 7x24 physical security monitoring?

One of the most important things, if you have a nontrivial computer room, is to consult a local expert who can advise you on what is the most appropriate protection in your area and for your situation.

How can you attempt to recover from physical damage? The classic answer is, of course, to have a redundant offsite installation that can be used for recovery (see "Redundant Premises" below). Alternatively, consider such options as

- emergency recovery agreements with key suppliers
- strategically selected and located spares
- planning for what processes and activities (if any) can be performed manually while computing system recovery is under way

Utility Failures

Just about everyone is in a position to be affected by some form of utility failure, the most obvious being electrical power failure. Even if you generate your own electricity with wind turbines and backup batteries, you're still at risk of extended calm or physical failure of your generating equipment. Most people can survive short outages on an occasional basis. But if you're in an area where utilities can be unreliable (poor infrastructure, frequent thunderstorms) or if you worry about extended outages such as those suffered in Quebec and New England this winter due to the ice storms (some places were without electricity for several weeks), you may wish to consider some suitable form of backup or redundancy for your utilities.

Electricity

Most of us rely on electricity from the local power company. The obvious way to protect yourself against outages is through the use of an uninterruptible power supply (UPS) with a diesel generator for backup and extended outages.

But it's important to remember that, in a disaster, you won't be worried only about your central computing systems. You'll need power to run heating or cooling equipment, ventilation, at least some room lighting, your telephone switch, and so on. This isn't just a system administration issue – it's a facilities-wide issue.

Water

From a system administration perspective, the primary use for water is in air conditioning equipment. In some situations, a reservoir or cistern could provide spare water dur-

ing an outage, and water tanker trucks are sometimes available. Otherwise, hope for a cool spell.

Gas, Oil, Propane

Again, these are used primarily for environmental control. Fortunately, alternate heat sources are often available, even if you have to resort to electric space heaters.

Communication Links

Most of us rely on some form of communications, whether it's ordinary telephone connections, leased lines, fiber, or various forms of wireless communications (though it's probably safe to say that wireless use is in the minority). Most of us rely on these links as a regular part of our workday, and for some of us, the business stops when the communication links go down.

The best way to protect your communication links is through the use of redundant connections. For Internet connectivity, many organizations are "dual homed" to two providers, and prudent organizations make a point of ordering links from multiple carriers (and hope that the carriers don't simply buy capacity from each other). Even if your redundant links leave your premises through different paths over different carriers, it's still possible for them to terminate in or pass through the same carrier central office, which does limit your redundancy. If you're provisioning multiple links, try to get specific physical routes from your carriers so that you'll have a better idea of where your exposures are.

One alternative for redundancy or backup that is becoming more common and more feasible is the use of metropolitan area wireless communications and/or satellite links. A satellite link, although typically slower and more expensive (or at least not cheap) provides nice redundancy because it can enable you to isolate your communications from any local problems. Of course, if all your connections are to systems in the same area as your office, remote satellite connectivity might not help too much. (And, of course, this is where I remind you all of the recent satellite outage that disabled huge numbers of pagers and the difficulty of dealing with failures in your backup communication systems.)

Physical Inaccessibility

I'll dredge up the ice storms from this past winter as an example of how your office can be fine, but it's just not possible to get there. Other examples are, of course, earthquake, flood, trucker blockades on European highways, bombs in the World Trade Center, and major parades. If your business relies on physical access (e.g., a printing company, a warehousing company, etc.), you could be in trouble. If you're an organization that deals in knowledge or computing, you might be better off. An easy way to deal with the latter is to ensure that your staff has home computing and an account on a reliable ISP (or run your own remote access servers, with lots of capacity for emergencies), and just have them dial in for the duration. Voice mail, remote phone forwarding, cell phones, and pagers all help limit (if you're lucky) the impact of this kind of disaster.

Redundant Premises

The classic disaster recovery plan (for computing and communications, at least) involves a redundant recovery site, just sitting and waiting for something to go wrong. This is still common in the mainframe world, where large data-processing capacity is needed on an ongoing basis. If you absolutely need ongoing computing, an alternate site is likely going to be part of your plan. Even if your needs are much simpler, you can benefit from some forms of offsite redundancy.

If you absolutely need ongoing computing, an alternate site is likely going to be part of your plan. Even if your needs are much simpler, you can benefit from some forms of offsite redundancy.

The key to successful recovery is a proper plan and proper documentation.

Standby Sites

In the traditional case, a large, climate-controlled, raised-floor computing center is loaded up with millions of dollars of equipment and sits there idle waiting for something to go wrong somewhere else. These sites have taken a number of forms. One of the most common is to be run by a service company, providing backup services to a number of clients. But some large organizations have backup computing centers that are dedicated to them. It is also not unheard of for system vendors to offer recovery services for their customers, and some cooperative ventures also exist for their members' mutual benefit. This typically isn't a cheap method of protection, but if you need it, you need it.

Distributed Sites

What is much more feasible, and much more practical in these days of high-speed Internet connectivity, is the use of distributed computing sites, which can provide backup for each other in the event of a disaster. An obvious example is the use of Web server hosting at service providers using some form of load sharing across multiple servers. This can be expanded by distributing your primary computing resources across multiple sites, taking care that you have similar equipment at each site. This kind of distribution also makes it possible to automatically store your backups offsite (given large enough bandwidth). However, it is harder to justify distributing your computing when your staff is all located in one place.

Recovering

I mentioned it before, as many others have mentioned before me, but I'll reiterate that the key to successful recovery is a proper plan and proper documentation. Space limits restrict how much I can say here, and I will refer you to the bookstores for DRP books, but I will mention a few points.

- Hardware. Is your recovery hardware compatible? Do you have documentation of what makes your systems and installations unique?
- Backups. You have them, of course, and they are offsite, of course. But do you have the index to the media that will allow you to find the necessary backups when you need them?
- Names and addresses. Do you know what your machines should be named and numbered?
- People. Do you know how to get in touch with your staff at home, or is your only phone list the office numbers on your desktop machine that was just destroyed in the fire? Do you have a list of who to contact first, who should do what, in what order?
- Communications and connectivity. Do you know whom to call at your service providers and carriers to get your connectivity adjusted for your new or temporary location?
- Status. Do you know who is in charge of the entire recovery operation for your organization and how to get in touch with them and when?
- Food. And finally, do you have a list of food delivery places at your recovery site so that you won't pass out from lack of sustenance while working feverishly to put things back together?

Summary

In summary, plan, prepare, and pray you never need it.

loading source code UNIX on the PC

By Bob Gray

Bob Gray is co-founder of Boulder Labs, a digital video company. Designing architectures for performance has been his focus since he built an image processing system on UNIX in the late 1970s. He has a PhD in computer science from the University of Colorado.

<bob@boulderlabs.com>

My first article provided motivation for running source code UNIX (SCU) on your system (April 1998). The second article (June 1998) was devoted to evaluating and choosing hardware. (Both are online at <www.usenix.org/publications/login> and <www.boulderlabs.com>). By now, you have some equipment ready for a system and you have chosen your software. This article walks you through some of the issues you will face when loading the software. You are expected to be familiar with PC hardware and general UNIX system administration. Of course, the process varies between loading Linux, FreeBSD, NetBSD, OpenBSD, and BSDI. All CD-ROM distribution sets come with instructions; this article gives general guidelines. You'll need to consult the specific references but overall the required steps for each are similar.

Where are you going to load your source code UNIX? You'll want a few hundred megabytes for a basic X11 development system and a gigabyte or more if you install lots of applications (also known as "ports" or "packages") and their source code. Are you going to share a disk among multiple operating systems, or can you dedicate your disk to one OS?

My system provides an example of these issues. I have a 2GB disk that is capable of booting either Windows 95 (for Word and Excel) or FreeBSD. I only occasionally need a Win95 environment, so I gave it 500MB of the disk. The FreeBSD system got 1,500MB. When I power on the system, I get a boot screen that gives me two seconds to explicitly choose an OS; otherwise it boots UNIX by default. Many of the SCU systems understand the DOS filesystem. I "mount" the 500MB partition and therefore have an easy mechanism to move files between operating systems. (I also use FTP to transfer files between another networked workstation.)

Although it is possible to load the system from the Internet, this article works with a CD-ROM distribution set. CDs are cheap. A set has tons of other useful software. If you have problems, it is easy to start over if you have the media. Most of the distributions come with some instructions; FreeBSD has a booklet, OpenLinux provides a manual, etc. Generally, the CDs themselves contain documentation – sometimes as text files and sometimes as html files. You can use a running system to look at these files. Each system's home Web site also provides instructions, FAQs (frequently asked questions), manual pages, and search engines to look up issues. There are also many books published to give you guidance, including *The Linux Bible*, *The Complete FreeBSD*, and one I recommend for all systems, *UNIX System Administration Handbook* by Nemeth et al. It would be wise to become familiar with these resources before you begin. It is also handy to have these references available while loading the system. This means another working system with Web access.

The basic process for all SCU systems on PCs is:

1. Find some disk space
2. Boot an installer program
3. Partition your disk
4. Install a boot manager
5. Choose software that you want installed
6. Install from CD
7. Perform post install configuration
8. Boot your new system, configure and administer it

References:
<www.freebsd.org>
<www.linux.org>
<www.netbsd.org>
<www.openbsd.org>
<www.bsdi.com>

Thanks to Tom Poindexter and Mike Durian for reviewing this article.

Under control of the installer program, you'll have to make decisions about the disk layout. Unfortunately, the word "partition" is heavily overloaded.

Step 1. Finding Disk Space

If you have a new disk that you are going to dedicate to SCU, jump to step 2. If you have important stuff on your disk, *back it up* and verify your backups. If you have an existing Win95 system spanning the whole disk, you can coalesce the files using the DOS 6.xx DEFRAG utility or Norton Disk tools. This will bring all the used blocks to the front of the disk. Then the FIPS utility (provided by many SCUs on the CD) can be used to split the disk into more than one partition. Thereafter, Win95 will have to live within its new bounds, and the new partition(s) will be available for SCU. Another possibility is having two disks and an operating system on each. The boot manager allows you to switch between them.

Step 2. Booting an Installer Program

Now it's time to boot your SCU's installer program. There are several possible ways:

1. With the right CDROM controller (supports "El Torrito") and the right SCU (FreeBSD for one), you can boot the CD directly.
2. On some SCUs, the installer program can be booted from the CD under DOS.
3. Most versions allow you to boot the installer from floppy. Either the floppy is provided or you can copy an image from CD to floppy using a program (`rawrite.exe`) under DOS or using `dd` under UNIX.

Your installer program can be quite impressive. On a little 1.44MB floppy, you have a kernel capable of dealing with tons of hardware combinations. It also contains a user interface that must support the spectrum from novice installers to expert power users. Most installer programs are capable of loading from CD-ROM, a hard disk, or a TCP connection. Some programs include significant online documentation to help with the loading process. Some SCU systems use a second floppy as a `/root` disk during the install process.

One crucial issue for your installer program is that it recognize the relevant hardware pieces of your system. For example, you may have the brand-new Intel DK440XL motherboard that has the new Adaptec AIC7895 SCSI chip and 10/100 Ethernet chip onboard. If your disk and CD-ROM drive are attached to this built-in bus, most existing installer programs (through June 1998) will not recognize the hardware. There are several ways around this. You could use IDE drives to load the system, then build a kernel that can handle the new hardware. You could buy or borrow a SCSI PCI board that is recognized. Or you could have someone build an installer program that knows about the new hardware.

Step 3. Partitioning Your Disk

Under control of the installer program, you'll have to make decisions about the disk layout. Unfortunately, the word "partition" is heavily overloaded. When talking about DOS and BIOS, there can be up to four primary partitions on the disk. One is marked as the "active" partition – from which the boot will be attempted.

FreeBSD calls a DOS partition a "slice." A FreeBSD partition is most like the historic UNIX partitions, where each disk drive had a label indicating how the disk was laid out. For many UNIX systems including FreeBSD, partition `a` is conventionally the root, partition `b` is swap, partition `c` is the whole disk, and partitions `d-h` are optionally used for filesystems such as `/usr`, `/var`, or `/tmp`. The `c` partition overlaps all other partitions. The `disklabel` command establishes the mapping between partitions and disk blocks within a slice (`/etc/fstab` and the mount command control the mapping

between filesystems and disk partitions). Under FreeBSD, each slice can contain a full set of partitions. A scheme for naming each partition on SCSI drive 0 therefore is:

```
/dev/sd0s{slice-number}{partition}
```

where `slice-number` is 1-4 and `partition` is a-h (e.g., `/dev/rsd0s4a` , for drive 0, slice 4, partition a)

If you don't need to run multiple operating systems on a single disk, you can ignore most of this slice stuff with FreeBSD. See `fdisk` for details.

Linux uses a different scheme. If you don't need any more than four DOS partitions, you use these directly. For example, `/dev/hda1`, `/dev/hda2`, `/dev/hda3`, `/dev/hda4` for the disk called Hard Drive A. As an example, you might have

`/dev/hda1`: Windows 95
`/dev/hda2`: Linux root file system
`/dev/hda3`: Linux swap partition
`/dev/hda4`: Linux /usr mounted filesystem

In cases where more partitions are needed, Linux uses DOS "extended" sub-partitions. For an example of four operating systems on a single disk, see:
<www.linuxresources.com/LDP/HOWTO/mini/Linux+DOS+Win95+OS2-1.html>
<www.freebsd.org/FAQ/FAQ105.html#105>

Step 4. Installing a Boot Manager

The job of the boot manager is to load the operating system and begin its execution. This simple task is complicated by different kinds and geometries of disks, multiple IDE and SCSI controllers, various BIOSs, and the possibility of multiple operating systems on a disk. When booting, BIOS loads the first block of the disk (called the Master Boot Record or MBR) and, if it has the correct magic numbers, jumps into it. The MBR code looks at the partition table that is embedded within it to determine which partition to boot from. Then that code is loaded and executed. Soon a boot manager is running (i.e., it is capable of choosing and loading the operating system).

Here is what the FreeBSD default boot manager looks like. You can see the large matrix of possibilities.

```
Usage: bios_drive:interface(unit,partition)kernel_name options
bios_drive   0, 1, ...
interface    fd, wd or sd
unit         0, 1, ...
partition    a, c, ...
kernel_name name of kernel, or ? for list of files in root directory
options      ...
```

Examples:

```
1:sd(0,a)mykernel boot `mykernel' on the first SCSI drive when one
      IDE drive is present
1:wd(2,a)      boot from the second (secondary master) IDE drive ...
```

Other boot managers include LILO and OS-BS. LILO or LInux LOader is a limited boot manager that is often installed by default on Linux systems. OS-BS gives you more control over the booting process than the default boot manager, with the ability to set the default partition to boot and the booting timeout.

Here is a procedure for having FreeBSD and Windows on a disk. Windows 95 is picky about where and how it is placed on the hard disk – it need to be on the first primary partition on the first hard disk. You need to install Windows 95 first, then FreeBSD.

The job of the boot manager is to load the operating system and begin its execution. This simple task is complicated by different kinds and geometries of disks, multiple IDE and SCSI controllers, various BIOSs, and the possibility of multiple operating systems on a disk.

FreeBSD's boot manager will then manage to boot either Win95 or FreeBSD. If you install Windows 95 second, the process will overwrite your boot manager without even asking. Then only Windows will boot.

Warning: depending on how your BIOS is configured, you may not be able to boot operating systems that are in partitions more than 528MB from the start of the disk. A simple workaround is to boot the boot manager from floppy. It doesn't have the 528MB limit and can load an operating system from anywhere on the disk. For more information on boot managers, a discussion of this limitation, and some other solutions, see the following Web reference: <www.freebsd.org/tutorials/multios/multios.html>. (P.S. It's a good idea to have a boot floppy around anyway for disasters.)

Step 5. Choosing Software to Install

The installer program will give you choices as to what software will be loaded onto your disk from the CD. Some systems will allow you to select from broad categories. For example, you might be asked to choose one of the following:

- minimal system
- minimal system with kernel sources
- development system
- development system with X11
- custom
- everything

Later, with a package manager, you will be able to adjust what is on your disk by adding or deleting packages.

Other installer programs present you with a list where you can toggle on or off the desired subsystems. For example, with Red Hat Linux, you set * for the pieces you want installed:

```
[ ] Printer Support
[*] X Window System
[*] Mail/WWW/News Tools
[ ] DOS/Windows Connectivity
[ ] X Games
[*] Networked Workstation
[*] Dialup Workstation
[ ] News Server
[ ] NFS Server
[ ] DNS Server
[ ] Samba Server
 ...
[ ] C Development
[ ] Everything
```

Step 6. Installing from CD

At this point, you will have specified that the distribution is on CD-ROM and you are ready to have the installer copy material onto your disk. This step takes from a few minutes with a fast CD-ROM and minimal system to more than an hour. Most installers provide some kind of progress monitoring and logging.

Step 7. Performing Postinstall Configuration

It should be possible now to boot your system and configure it. However, it may be more convenient to let the installer program ask you a bunch of questions and then configure the system.

If you will be running X11, you should know what kind of graphics card you have, the capabilities of your monitor, and your type of mouse. If your graphics card is not listed in the options, you may be able to get by with a generic SVGA X11 server. My previous article gave some references on finding software for the latest graphics cards. Be warned, configuring X11 can be tricky. There are a lot of parameters that have to be right for it to work.

You'll probably have some kind of network connectivity, so you'll need a hostname, an IP number and mask, a default gateway, a domain name, and a DNS server. Hope that there were no IRQ conflicts and your installer program just cleanly found your network interface card. If not, the various systems have ways of giving the software probing hints.

Often you can set the time zone and the root password from the installer program. You may also have the options of selecting what daemons get started at boot time, for example, `cron`, `nfs`, `network`, `sendmail`, `syslog`, etc.

Step 8. Booting, Configuring, and Administering Your New System

You've made it. Boot the system. Now it's just a matter of bringing in your personal environment and loading more ports or packages. You may want to add new users. You will have to configure your servers. You will want to configure and build a custom kernel and run it instead of the "generic" kernel. Not only does this make the kernel smaller, but it allows you to add nonstandard hardware drivers and other kernel features. Your boot time will be shorter if your kernel doesn't waste time probing for nonexistent hardware. You'll want to size your kernel and select various options such as NFS, DOS, and/or shared memory capability. Then you will describe your exact hardware, only these drivers will be built into the kernel. Make the kernel, install it in the root, and reboot.

You will want to configure and build a custom kernel and run it instead of the "generic" kernel.

In Case of Trouble

Sometimes just going through the procedure more than once will "fix" a problem. Because you know what to expect during the install, you are better prepared to have the "right" answers ready. I once got stuck in a partitioning menu and couldn't get what I needed. Something I had done earlier was inappropriate. I came back later after a fresh reboot, and everything worked fine. I guess the installer program was not bulletproof.

If you need to revert the MBR back to Win 95, you can boot a DOS floppy and type:

```
fdisk /mbr
```

Find a knowledgeable friend or colleague to help you with installation problems. Go to the Web sites and use their search engines to look up questions. Try posting well thought-out, detailed questions on USENET. Be sure to include relevant details so that those folks have enough information to help you.

Sometimes the way to get over a hump is to switch to a different SCU. I had problems with one distribution of Linux recognizing some hardware, but the InfoMagic Red Hat version worked fine.

Find the experts for your hardware. Their SCUs usually have the best driver support. So if you are running a multiprocessor system, read the newsgroups to find the developers. If you need great fast Ethernet support, you'll find the best driver/harware combinations. It's common for one person to write code for new hardware under his or her SCU – then the others import and incorporate this driver some time later.

The next article will be dedicated to application packages and how to manage them.

small tools for automatic text generation

by Diomidis Spinellis

Diomidis Spinellis holds a PhD in computer science. Currently he is leading software development at SENA S.A. His research interests include information security, software engineering, and programming languages.

<dds@senanet.com>

The Problem

The textual description and analysis of large data sets can be a repetitive and error-prone task. Examples of data sets that may need to be textually described include population statistics, market research results, scientific and engineering data such as chemical compounds or earthquake structural damages, and literature surveys.

In all such cases the data need to be organized and presented in a meaningful way. In some cases this process is periodically repeated (e.g., market research results); in other cases the base data set is frequently updated (e.g., a literature survey). Automating the data presentation process can reduce the work required to generate the reports, eliminate a source of errors, enhance the outcome's consistency, and speed up the generation process.

In the following paragraphs I describe a small set of special-purpose tools developed for the automatic production of a multiparadigm language literature survey. A set of 104 languages combining different programming paradigms needed to be presented in a meaningful way, tabulated, indexed, and cross-referenced. I decided to divide them according to the paradigms they supported and present them by listing certain characteristics of each language. During the course of my investigation the set of languages surveyed grew by 100% and was frequently revised; this made the hand editing of the text error prone and unproductive.

Although the tools developed are special purpose, the methods used to construct the tools and generate the output can be applied in a number of different situations.

Functional Description

The results of the survey were entered into a simple database structured as a text file. The file is structured similar to a *refer* database: records are separated by empty lines, and record fields are identified by a letter following the percent character at the beginning of a line. Figure 1 shows a sample record from the database. The record's fields are the name of the language *N*, significant references *R*, its characteristics *C*, its usual implementation *I*, the paradigms it supports *P*, and a short descriptive text *D*.

```
%N Modula-Prolog
%R Mul86
%C UN RL RT nDT BR (SLD,EX) BR nRT
%I Run-time library for Modula-2
%P Logic Imperative
%D
The facilities of a Prolog interpreter are provided to a Modula-2 programmer
through a library.  Predicates, which can be called from the Prolog inter-
preter, are written in Modula-2. The library includes term-handling proce-
dures.
```

Figure 1. A sample record

The text generator scans the database and divides the languages into categories according to the paradigms supported by each language. Each such category (e.g., languages that support the logic and object-oriented paradigms) is formatted as a separate text section. The generator formats the title as in the following example:

2.2.5 Combinations of Imperative and Logic Paradigms

It then inserts some manually prepared descriptive text and appends text that tells the reader the number of languages in that category, provides pointers to the summarizing tables (examples are Tables 1 and 2), and introduces the paragraphs to follow. The following is an example of the automatically generated text:

> We found ten languages that combine the imperative and logic programming paradigms. Their implementations are summarized in Table 1 and their characteristics in Table 2. In the following paragraphs we list the most important features of each language.

Name	References	Implementation
2.PAK	[Mel75]	Language
C with Rule Extensions	[MS90]	Extension of C, preprocessor
Leda	[Bud91]	Language
Logicon	[LC86]	Prolog interpreter in Icon
...		

Table 1: Implementations combining the imperative and logic paradigms

Name	Characteristics						Control
	BR	||	R	RT	U	DT RT	
2.PAK	✔				✔	✔ ✔	*
C with Rule Extensions	✔		✔	✔	✔	✔ ✔	*, X
Leda	✔		✔	✔	✔	✔ ✔	SLD, X
Logicon	✔		✔	✔	✔	✔ ✔	X, SLD
...							

Table 2: Characteristics of imperative and logic paradigm combinations

A description of the languages based on the *R*, *D*, and *N* record fields is then produced:

> **2.PAK** [Mel75] Block structured language offering user-defined pattern matching and backtracking.
>
> **C with Rule Extensions** [MS90] Based on the C programming language with an extended syntax, a richer set of data types, a flexible input/output system, and a forward chaining [Ric83, p. 56] execution strategy.
>
> **Leda** [Bud91] Language with syntax similar to that of Pascal, with an additional code abstraction facility, the relation. The data-space for all entities contains the undefined value. Relations are coded as Prolog rules and allow backtracking.
>
> ...

The generator inserts at the end of each section another tailor-made paragraph containing concluding remarks on that section. A special section contains all languages that could not be fitted into one of the preceding sections together with a special table listing the paradigms each language supports (Table 3). After all sections have been processed the generator produces a summary of the number of languages supporting each paradigm combination (Table 4).

Name	Paradigms
DSM	Imperative, Object-oriented, and Relational
Echidna	Constraint, Logic, and Object-oriented
Educe	Database and Logic
Enhanced C	Imperative and Set
Fooplog	Functional, Logic, and Object-oriented
Icon	Generators and Imperative
KE88	Functional, Logic, and Object-oriented
Kaleidoscope	Constraint, Imperative, and Object-oriented
Lex	Imperative and Regular-Expression
ML-Lex	Functional and Regular-Expression
...	

Table 3: Languages and paradigms they support

Functional	●		●	●			●		●
Imperative		●	●			●	●		
Object-Oriented				●	●	●	●	●	
Logic	●	●			●		●	●	●
Distributed								●	
Constraint									●
Number of languages	24	10	9	8	11	7	5	4	5

Table 4: Summary example

Implementation

I implemented the generator as a set of ten small tools using a separate program to administer the database in cases where a text editor was not adequate. Each of the ten tools performs a small specialized task. The tools are implemented in the Perl and Bourne shell (`sh`) languages, making use of additional UNIX tools such as `grep` and `sed`.

The system's driver is the `maketext` program, which, given a list of interesting paradigm combinations, generates the section title and the opening paragraph. It then calls the external program's desclist to create the description list and chartable/imptable to create the characteristics and implementation tables.

After the whole database has been processed, the same program generates the summary table. One other program, partable, generates the language/paradigm table for the last catch-all section. All programs take as an argument the paradigm combination described in the section processed, or (for the last section) the paradigm combinations already processed. `Maketext` is implemented in Perl, making extensive use of regular expressions to parse the database.

The second-level programs operate on the output of the `pars` program, which, given a paradigm combination, generates a sorted list of all records exactly matching that paradigm combination. A similar program, chars, generates only the characteristics of the paradigm combination.

`Pars` depends on its operation on the `dbgrep` program, which scans the database for records matching the search criterion specified as an expression possibly containing string regular expressions. `Dbgrep` also takes a flag to display all records not matching the criterion. This feature is used in the generation of the last section.

In order to match the record output order of these programs and create meaningful tables, the records and fields are sorted in various phases of the text generation, using either the sort statement available in Perl or two additional programs, `linesort` and `llinesort`, which sort the words on every input line. `Llinesort` performs this operation only on lines matching a specific pattern.

This division among the many specialized programs resulted in a modest implementation effort: 697 lines of code divided as indicated in Table 5. The largest program, `maketext`, is only 117 lines long including the text that is copied verbatim to the output file.

The output of the text generator is in the LaTeX text markup language. The 1134 line language database is converted into 1464 lines of LaTex, which also include commands to include another 371 lines of human-generated text.

Program	Implementation	Lines
chars	Sh	13
chartabl	Perl	59
dbgrep	Perl	27
desclist	Perl	17
imptable	Perl	38
linesort	Perl	19
llinesor	Perl	26
maketext	Perl	117
pars	Sh	13
partable	Perl	27
Total	Perl	697

Table 5: Implementation effort and details

Conclusions

The database described in the previous sections and the associated tools were used for almost three years. During that time, the database was frequently edited and revised. In some cases the output format was also modified. Both types of changes would have been very difficult if the data had been statically embedded in a document. The approach I described was used to change the structured database, and a single command reflected them in the camera-ready output. The ease of adding new records and modifying existing ones encouraged me to keep the database current.

I have thus found that special-purpose throw-away tools can be effectively used to generate text automatically from structured data collections. For those who might be interested to use this approach in other domains, the following suggestions may be of help:

- The use of a markup language such as HTML, troff, or TeX as tool output can be employed to focus on the problem to be solved and avoid issues of output formatting and device support. Troff preprocessors such as eqn and tbl have successfully employed this method.
- Human and generator-produced text can be intermixed to create more lively output and avoid the dry feel of generator output. Small tricks such as correctly spelling out numerals and adding a final "and" to a generated list also help.
- The leverage provided by the string handling, regular expressions, associative arrays, and metaexpression evaluation capabilities of the Perl programming language was invaluable. The idea of the automatic generation of text may not have been practical without Perl as an implementation vehicle.
- Although Perl has a number of useful capabilities, the use of pipelines to divide the work among small units that process their input and produce some output in conjunction with the use of standard UNIX tools was also critical to the implementation of the system.

interview with John Stewart

John Stewart is director of systems engineering at startup Digital Island Inc. and occasional speaker and trainer at conferences. Rob Kolstad interviewed John electronically in early June 1998.

Rob: Seems like you've worked at several high-tech companies over the last few years. You started at NASA Ames?

John: I started in the NAS (Numerical Aerodynamic Simulation) Division at NASA Ames working for Computer Sciences Corporation (CSC). My supervisor was Michele Crabb, who led the Distributed Software team that ported software from Sun to SGI to Cray to TMC to Amdahl and to anything else that needed it. DSS also supported the email systems and the netnews systems for that division.

I was at NASA Ames for two and a half years, working for two government contractors – CSC and Sterling Software. (NASA/NAS finished one contract with CSC and offered it up for bid. Sterling won that bid, so we all changed employers.)

In many areas, the government bids out support and development areas to private companies. The contract will often be multiyeared and have a fixed amount of money associated with it. I worked for CSC/Sterling, and CSC/Sterling was contractually obligated to work for NASA Ames, NAS Division to provide the work we did as employees.

Rob: And then you made your move to Cisco as a full-time employee. How did you choose Cisco? What were your responsibilities there?

John: Cisco was a complete surprise, and I made one of the hardest decisions I've ever made in my career. I was originally leaving NASA/Sterling to work at QMS (network/systems administration for a printer company) – I needed a change of scenery from NASA, and Hal Pomeranz mentioned QMS was looking. A while before talking with QMS, I had dropped my resume electronically to Cisco, which called me the day before I left NASA Ames.

I met, that night, over dinner with the hiring manager at Cisco. He made an offer the next day (a Friday), and I didn't sleep much at all the entire weekend.

Cisco was hiring a senior systems administrator to support the Technical Assistance Center (TAC). The systems administration team actually reported into Advanced Customer Systems, the team responsible for developing CIO (or CCO, as it is now known, the Cisco external Web site that has been its flagship for electronic customer support and e-commerce). I ultimately worked with that team, too, because I had background doing Web-based application development. So I was soon to lead a dual life.

I had committed to work for QMS, but Cisco was a real opportunity with the up-and-coming networking company.

On Monday, I walked into QMS, sat down with the HR manager and the VP of the division, and told them what I felt I had to do for myself and, to some degree, for them. I would have regretted turning down the Cisco offer if I did, and that would have made my tenure at QMS very short. QMS was really understanding, and I left for Cisco.

Rob: Do they work you hard at Cisco? How many hours were you putting in for a typical week?

John: It went up and down. I didn't record it closely, but I am betting it was 60 hours a week on average. I had a great team of people that I was working with, both in the developer part as well as the systems administration part of my job. Cisco demands you respond when things are on tight deadline or broken; at the same time, my management was more than willing to reward hard work and give time off when the deadline or crisis was over. It was a great place to work and wasn't easy to leave.My time off was critical in having a family life, seeing my sister and her husband (who lived ten minutes away when I worked at Cisco), and travelling. I learned the first year I did it that a two-week vacation in August going to places where the most advanced technology is rotary-

dial telephones (upstate Vermont on a small lake) is crucial to avoiding burnout. The mountains, swimming, hiking, and just lying around for a few days really help me avoid burning out completely.

Rob: Lots of people say "60 hours per week." I think many of them include lunches, dinners, and lots of other things in their tally. Did you really work 60 hours/week?

John: I was in at 7:00 am, out at 7:00 pm, and working on weekends. We also had downtime, outages, travel, weekend projects, etc. I didn't eat lunch most days and didn't eat dinner until late.

Rob: And then the startup fever hit. Please tell us the story.

John: I left Cisco because the company wasn't the same as it used to be; part of the culture was changing. Cisco can't be the same company I joined because, when I left, there were over 10,000 more people in it than when I joined. I also left because, over time, the team I joined and wanted to learn from had migrated out as well. Only one person remained in advanced customer systems who had been there when I joined just two years prior.

Digital Island's history, and my association with it, is 100% Silicon Valley classic. Ron Higgins (CEO) had the original idea to build this international private network. He recognized he had extensive expertise in building a company, but not the technology expertise to build the network he wanted. Through a friend, he met Allan Leinwand (now DI's CTO and formerly one of Cisco's top international network design engineers).

Allan and Ron designed the entire network on napkins in a San Francisco restaurant. Allan became a consultant to DI at that point and joined full-time in early 1997. Allan Leinwand knew Bruce Pinsky (now DI's CIO, but at that time top customer support engineer at Cisco, and arguably the best network diagnostic technician in the world). Bruce knew networking and some systems support as well. Bruce wanted a person who built scalable support environments on UNIX-based platforms, which is what I helped do at Cisco. Bruce was one of the internal customers I supported and a friend, as well. Bruce recommended me, and I started consulting for DI, too.

I wasn't recruited though, really. I asked the CEO of Digital Island (Ron Higgins) if I could switch from being a consultant to full-time, and he offered me a full-time position. By then, all seven of the original people had left their full-time jobs and were full-time at DI; I was the last holdout. It wasn't easy leaving Cisco, but I don't regret it.

DI was an easy choice because there was never a downside. I didn't take a salary hit; I was enticed by great stock benefits; and I already knew who was involved (and these are top people in the network design industry). I feel good about helping build a company from the ground floor up along with being surrounded by top-flight people in their respective areas from our sales team all the way to our board of directors. As I've heard from more than one person, startup companies just don't usually go this way.

I jumped from a big company to a small company at this point in my life because I am early in my career. The industry has plenty of opportunity right now, and it sure felt like the right time to try a startup company.

Rob: What kind of position do you have? What's a typical work day like?

John: A typical day for me starts at 5:00 am (alarm!) and I head in on the "boat commute." I arrive at the office around 7:00 am. By then, in most cases, I have already spent 30-60 minutes working, using my portable PC and cellphone. Getting into the office at the quiet time, which is 7:00 am, lets me try and catch up on email, paperwork, orders,

John Stewart

"I feel good about helping build a company from the ground floor up along with being surrounded by top-flight people in their respective areas from our sales team all the way to our board of directors. As I've heard from more than one person, startup companies just don't usually go this way."

etc. before jumping into the fray. As a technical manager, I spend half the day wandering around talking to people, listening, and reacting to ideas that they are working on, finding out what walls they are running into, and trying to break down some of those walls.

One thing I am definitely learning about the startup experience is that when you are going so fast, a lot of communications get scrambled. It is really hard for me, and for all of us, I think, to keep pace at this speed. Small miscommunications can send you down the wrong path so quickly, and under this type of pressure any wrong turn is magnified. You're working, you're under pressure, and having all the right information is key to doing the job efficiently.

I spend time working out how we're going to be arranged in our newly expanded office (we move in on Friday), ordering frame relay circuits for new employees, talking about product direction, and planning out people's time. Time management for myself and the team is critical and knowing when there just isn't enough time with the existing resources is one of things I keep a really close eye on. Sometimes the answer is just "no, we can't do that" and the reaction to that can be "we'll get you more people" or "reprioritize, this can slip."

This, and I'm still not doing techy stuff yet. I've been the product developer for a recently released product, participated in feature advancements for new products, and helped wherever I can. That is what everybody does at this company . . . fill in where they can. If it means sweeping the floor, sweep it. If it means figuring out if spending $250,000 in such and such a manner is the right thing, do it. You can't afford to say you are above or below doing anything – it is a startup.

I've been with the company for one and a half years now, working very long hours . . . even more than Cisco.

Rob: What's a "boat commute"?

John: About the time I left Cisco to join DI, I was also looking to buy a house in the Bay area, a tough and very pricey market. Rhonda (my wife) found a great town (Benicia) north of Digital Island's second home where I work, San Francisco. Benicia is just a short drive from a high-speed ferry service that heads in and out of San Francisco and has a stop in San Francisco only four blocks away from our office.

That makes my commute interesting, because it's about 1.25 hours total and includes sitting on a high-speed catamaran ferry cruising across the San Francisco Bay. I sit, read the newspaper, fill out expense reports, talk on the cellphone, which makes the ostensibly long commute turn into a piece of cake and actually helps me wind down at the end of the day.

Rob: It seems like, even with all this investment in work, you're also living a life. Do you have a family? Hobbies? Do you sleep?

John: I'm very happily married (for five years this week) and have a four-year-old son. Balancing time with them and time at work isn't easy; I wish I could be both places at once. Rhonda has always supported what I do, for which I can't thank her enough, and one of the things I am learning (not easily) is that, if you can't spend as much time doing something you want to do, better to make the time you do spend count.

When I'm at home, I am learning how not to be distracted by the computer and frame relay in the office. I'm learning that even if I am totally exhausted, playing a game of "Sorry" with my son needs to come first. The best part is that I can have a bad day, come home, and the first thing that happens is my son races up and hugs me. At that point, the bad day gets a whole lot better.

It isn't easy. Work could take up my entire day (and night!); sometimes I don't make that switch and sit down at home. I've literally asked Rhonda to make me realize I am doing that so I stop myself. She hates doing it; she thinks it is nagging. But sometimes I just won't see it myself and need her to remind me.

I get home at around 6:45 pm, having spent the ride home working, in most cases. In the evening, Rhonda and Jonathan become the priority, playing baseball outside, or going out to dinner, seeing friends or barbequing.

For longer periods of time where we can manage them, I really enjoy camping, having friends over, or going over to friends' houses and just socializing or riding my motorcycle. I'm a big fan of riding a motorcycle because it makes me concentrate 100% since I usually take it on twisty roads. It really forces work out of my mind because the distraction could create unwanted results.

Rob: Does Rhonda work outside the home?

John: For the first three-quarters of 1997, which is when I started at DI, Rhonda was working as a zookeep trainer at Marine World Africa. She left and decided not to go back, but it's funny – it's not because I was working at a startup, it was because the work schedule she had to adhere to included weekends, so we couldn't do anything as a family on weekends.

Nowadays, she is taking care of us (Jonathan and me). Because of the startup, she is really taking care of our home and our family – she manages it if you will. I'm glad for that. She has been the most supporting person in my life every time I've needed it . . . and I'm not just saying that.

Rob: I notice that you're also writing articles for *Webserver* and occasionally teaching tutorials at conferences. How do you work all that in?

John: Traveling means plane time. I don't sleep on planes much, even on overnight flights; so I read, write, and type on the laptop. I also have that boat ride on most days of the week, and I use that time to write articles, finish my course materials, or even play solitaire. I like writing; it's work but isn't "my job." Teaching is the same way. Teaching is something I have always enjoyed doing; I taught BASIC programming when I was in middle school. I just can't be a full-time teacher, but see it more as a creative outlet that I really enjoy doing and don't want to lose.

Rob: How much teaching do you do? How much time does course development take?

John: I teach between two to five times a year (lately, two) and speak at least one to three functions a year. Part of life is participating on the technical advisory board for Finjan Inc. – a company which founded the Java Security Alliance (JSA) and leads the active content desktop protection market. Speaking at the JSA events and participating at the events gives me another opportunity to talk and interact with industry members.

The course development takes time, no question about it, and sometimes I don't get enough to do 100% right, especially on a new course. It will take semidaily work over a two-week period, some days one hour and some six hours. The neat part about it, though, is that when you create it the first time, it takes a lot of effort; after that, you are refining it. Sometimes that will take as much effort, but usually you feel more confident, and the changes come easier, so then it takes a lot less. It is an evolutionary process, though.

Rob: Thanks for the chat. Best of luck at Digital Island.

John: Thanks, Rob. It has been quite a ride so far.

java performance

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

Optimizing String Processing

In the last column I started discussing some of the characteristics of Java performance. One of the issues mentioned was a method call overhead. In this article, I'll look at this a little more in the context of Java string processing and discuss how the immutability of strings affects performance.

Indexing Characters in a String

Suppose you have a need to find a character in a string, that is, return a position ≥ 0 of where a character occurs, or -1 if not found. In a language like C, with its pointers and low-level approach to programming, it's often just as efficient or more so to code a loop for finding a character directly instead of using a library function like `strchr()`. Especially with short strings, avoiding the function call overhead is often worth a lot.

Is this true of Java? Suppose we code up a similar example:

```
public class index {
  public static void main(String args[])
  {
   String s = "aaaaaaaaaa";
   int i = 10000000;
   int n = 0;

   // method #1

   if (args.length > 0 &&
     args[0].compareTo("index") == 0) {
    while (i-- > 0)
     n = s.indexOf('x');
   }

   // method #2

   else {
    while (i-- > 0) {
     int len = s.length();
     n = -1;
     for (int j = 0; j < len; j++) {
      if (s.charAt(j) == 'x') {
       n = j;
       break;
      }
     }
    }
   }
  }
}
```

When we run this code, we find that method #1 is about three times as fast as method #2, which is the equivalent of using pointers in C. Why is this? There are a couple of reasons. One is that `charAt()` is a method call, and not merely a quick peek that retrieves a character at a given position.

The other reason for the slower performance is that `charAt()` checks the index to ensure that it's within bounds. So even though we're iterating over the characters of a string in a safe way (`0` to `s.length()-1`), the checks are done anyway.

Because of the method call overhead and the index checking, `indexOf()` wins easily. It avoids the call overhead, and the index checking is not done, but instead characters are accessed directly from the internal `char[]` vector that underlies a string object.

It's probably a little early to say how an area like this one will shake out. `charAt()` could conceivably be expanded as an inline. The subscript checking is likely to be left in place because it's part of what Java guarantees to the programmer. And compilation to native code might change the characteristics of a string operation of this type.

String Immutability

Another aspect of string processing performance involves the immutability of strings. Suppose you want to create and print a list of numbers, like so:

```
public class immut1 {
  public static void main(String args[])
  {
   String s = "[";
   for (int i = 1; i <= 5000; i++) {
    if (i > 1)
    s += ",";
    s += Integer.toString(i, 10);
   }
   s += "]";
   System.out.println(s);
   }
}
```

This works okay, but seems kind of slow. When we profile the program, we find that it seems to be doing lots of data shuffling and so forth, with the garbage collector called repeatedly.

It turns out that using += on strings is quite expensive, and the reason is that strings themselves are immutable, that is, are not changed after creation. To append to a string, you must copy out the string to a StringBuffer, append to it, and then convert it back. StringBuffers are used for doing operations on strings, like + and +=.

This idea can be illustrated by the following example:

```
public class immut2 {
  public static void main(String args[])
  {
   String s = "aaa";     // sequence #1
   s += "bbb";
   System.out.println(s);
   String ss = "ccc";    // sequence #2
   StringBuffer sb = new StringBuffer();
   sb.append(ss);
   sb.append("ddd");
   String ss_save = ss;
   ss = sb.toString();
   System.out.println(ss_save);
   System.out.println(ss);
  }
}
```

These two sequences are equivalent; using += causes processing similar to that shown in sequence #2 (except for the `ss_save` line).

Note that we captured the old value of `ss` before `ss` was changed to point to a new string. The old string didn't change when we reassigned `ss`; we just changed a reference that pointed at it to point at a new string.

It turns out that using += on strings is quite expensive, and the reason is that strings themselves are immutable, that is, are not changed after creation.

Going back to the original example, we can rewrite it as:

```
public class immut3 {
  public static void main(String args[])
  {
   StringBuffer sb = new StringBuffer();
   sb.append("[");
   for (int i = 1; i <= 5000; i++) {
    if (i > 1)
     sb.append(",");
    sb.append(Integer.toString(i, 10));
   }
   sb.append("]");
   String s = sb.toString();
   System.out.println(s);
  }
}
```

resulting in a large increase in performance. A Java compiler is allowed to perform certain kinds of optimization on string concatenation operations, but keep in mind that string objects themselves do not change after creation (you can use StringBuffer for mutable strings).

Strings in Java are quite useful and powerful, and it's helpful to know some of their performance characteristics and bottlenecks.

using java

Is Java Secure?

Since its inception, the Java programming environment has been scrutinized widely with regard to its suitability as a secure environment. The result of this speculative approach (which it was in many cases) was that Java received less than a complimentary review in this context. Consequently, many corporate decisions were made based on the crystal ball approach, which was that Java was "insecure" and so was unsuited to applications such as electronic commerce. I have personally experienced this way of thinking.

The purpose of this article is not to venture unsolicited opinions on the decisions made in the corporate world regarding Java and its "secureness," but to objectively investigate the security model in Java. We will look at the "sandbox" model of security and look at some examples of code that will assist the reader to better understand how to translate a security "policy" into an actual "mechanism" to exercise that policy. Java security is an area of great interest to those developing virtual machines to third-party developers who write applications. I hope this article will shed some light on the Java security issues for all interested parties.

Java security comprises two parts: security inside the Java Virtual Machine (JVM) and security outside the JVM. We will look at security issues inside the JVM in this article. In subsequent articles, we will examine the security issues outside the JVM.

by Prithvi Rao

Prithvi Rao is the founder of Kiwilabs, which specializes in software engineering methodology and Java training. He has worked on the development of the MACH OS and a real-time version of MACH, and he holds two patents resulting from his work on mobile robots.

<prithvi@kiwilabs.com>

The work presented here is largely the research efforts of my brother, Ravi Rao, who also is the co-founder of Kiwilabs.

Security inside the JVM concerns itself with bytecode running on a JVM. The target of a Java compiler is the JVM and all bytecode is ultimately interpreted inside the JVM. The Java programming environment provides features that will supply mechanisms to ensure secure environments for running Java programs. In specific, we will examine the basic operation of the JVM, the classloader, and the security manager.

Examples of Java Security Violation

Java platforms have been successfully attacked in the past. The work by David Hopwood showed that the integrity of the JVM could be compromised by placing files in the cache by loading them as though they were from a local filesystem. Specifically Hopwood showed that, in early implementations of Java, when class names began with a "/", this permitted anyone who had access to a local filesystem to place "attack files" in those places. Also, it was shown that allowing access to a public FTP directory assisted in compromising the integrity of the JVM.

Another form of attack was applet based. In this scenario, a classloader was installed from an applet, which in turn installed Java classes as trusted classes. This was possible because of a bug in the Java bytecode verifier.

Language Support for Security

There are no pointers in Java but only references. The main difference is that with pointers you can do arithmetic that you cannot do with references. So it is not possible to scan memory.

Changing the cast of an object invokes a check at compile time or runtime. Any casting not done at compile time is done at runtime. Access to methods and fields are checked at runtime. If an application tries to access a method of a class that is declared as "private," a runtime exception is generated (`IllegalAccessException`).

The Java compiler produces a `.class` file that is neutral to architecture. It is therefore not possible to discover how a program is laid out in memory. This means that it is not possible to take advantage of any information about memory maps to devise attacks.

The Sandbox Model of Java Security

Java provides support to deal with two main problems, namely, unknown sources and immediate execution of bytecode. The term "sandbox" refers to the combination of classloader, bytecode verifier, and security manager to create a secure environment in which Java applications can run.

The bytecode is the part of the JVM that performs checks on the incoming bytecode. This process is transparent to users and it is not under their direct control.

The security manager enforces security policies for executable content. In other words, it controls the actions that a particular Java class can perform based on some policy.

The classloader is the connection between the JVM and the outside world.

How the Bytecode Verifier Works

Before a Java program can run, it must be loaded into the JVM. The loading process begins by reading a file that contains the bytecode (has a `.class` name) and storing this in a buffer. The verifier acts on the information in this buffer. It uses the fact that Java bytecode conforms to a well-known format (8-bit bytes) and that each Java `.class` file contains information about one Java class.

Java provides support to deal with two main problems, namely, unknown sources and immediate execution of bytecode.

Given that classloaders play a vital role in the loading of classes, the security manager must check to see if a class is allowed to create a classloader.

A `.class` file is a stream of 8-bit bytes, and larger quantities are represented in terms of 8-bit bytes. This information is also stored in big endian format (the most significant bits are leftmost). The first four bytes are the "magic number," which acts as an identifier. The other information contained refers to the major and minor number of the compiler that produced the `.class` file. The JVM is supposed to support classes that originate from a compiler with a fixed major number and a minor number that has a predefined upper limit.

The constant pool is an array of structures representing the names of classes, interfaces, fields, and methods. Debugging information is also available.

Verification is therefore part of the larger process undertaken by the JVM in order to execute bytecode. For instance, suppose we want to execute a "hello world" program. The following happens:

1. Load `.class` file into JVM.
2. Link bytecode with Java runtime.
3. Verify bytecode.
4. Prepare runtime (allocate resources).
5. Resolve references.
6. Initialize classes.
7. Invoke "main" method.

In all of this the main accomplishments of the bytecode verifier are:

1. checking of `.class` file format
2. protection against version skew
3. checking for stack overflow
4. checking for illegal data conversions
5. checking that instructions have proper parameters on the stack

The Classloader

The classloader is the link between the outside world and the JVM. All bytecode brought into the JVM must be done so under the auspices of a classloader. There is a default classloader as part of the JDK. Users who write their own may eventually call the default classloader if they are unable to load a class using other classloaders.

One of the requirements of writing a Java program is that users set an environment variable known as `CLASSPATH`. This variable is used by the default classloader to load "trusted" classes. The logic is that if there are classes found under `CLASSPATH`, then they must have been put there by the person who set this variable, so the default classloader can "trust" the class.

The alternative is that if there is a class not in `CLASSPATH`, a separate classloader must be provided to load it. The implication here is that the classloader is part of the identity of the class. For instance, browsers often use different classloaders to load classes from different sources. Given that classloaders play a vital role in the loading of classes, the security manager must check to see if a class is allowed to create a classloader.

In summary, two classes are of the same type only if they have the same fully qualified name (FQN) and they are loaded by the same classloader.

The class `java.lang.classloader` is an abstract class from which other classloaders can be subclassed.

```
protected abstract class loadClass(String name, boolean resolve)
throws ClassNotFoundException;
```

The following is an example of a classloader:

```
import Java.io.*;
import Java.net.*;
public final class URLClassLoader extends Classloader
{
Extend Java.lang.ClassLoader which is an abstract class
private String urlAsString;
```

This is a string containing the location from which this classloader will load files. It could be a URL such as <http://www.foobar.edu>. It is set once at the time this class is instantiated.

```
protected URLClassLoader() throws MalformedException
{
  this(null);
}
```

The constructor for this class takes no arguments.

```
public URLClassLoader(String urlStr) throws MalformedURLException
{
  if (urlStr == null || urlStr.length() == 0)
    throw MalformedURLException("No url provided.");
  urlAsString = urlStr;
}
```

This constructor just checks that there is a string and it has nonzero length. If not, then it throws a `MalformedURLException`.

```
public synchronized Class loadClass(String name, boolean resolve)
  throws Java.lang.ClassNotFoundException
{
}
```

Loadclass is the abstract method that must be implemented.

```
private byte[] readClassFile(String classFileName) throws
  FileNotFoundException, IOException
{
}
```

Note: It is beyond the scope of this article to provide the bodies for these methods. You can send mail to <prithvi@kiwilabs.com> for the code examples.

In summary the classloader does the following:

1. takes a name and produces a Class object
2. subclasses from `Java.lang.Classloader` (an abstract class)
3. defines method `loadClass` after extending ClassLoader
4. maintains separate namespaces

The Security Manager

The security manager is responsible for enforcing the security policies on Java programs. It controls access to resources on the platform such as filesystems and networks, environments on the host, and such things as the creation of processes. Most important, it is used to control the creation of classloaders.

The method `System.setSecurityManager()` is used to accomplish this goal. The class `Java.lang.SecurityManager` is an abstract class. The user must extend this class to implement a security policy. An example of part of a security manager follows.

The security manager is responsible for enforcing the security policies on Java programs. . . . Most important, it is used to control the creation of classloaders.

```
public class URLMain throws MalformedURLException,
      ClassNotFoundException
{
  SampleSM sem;
```

The class name we will use for our security manager is `SampleSM`. So we declare it to be of type `SampleSM`.

```
  Hashtable clHashtable = new Hashtable();
  URLClassLoader urlcl;
  Class cl;
  StringBuffer urlSB;
  StringBuffer classFileSB;
  int ch;
  String urlString;
  String classFileName;
```

Hash table to keep track of all classloaders and string buffers for reading URLs and class files.

```
  ssm = new SampleSM(true);
  System.setSecurityManager(ssm);
  ...
}
```

NOTE: The rest is beyond the scope of this article. If you wish to see the rest of this example, send mail to <prithvi@kiwilabs.com>

In summary:

1. Use Java.lang.SecurityManager to set security policy.
2. For each resource a check is performed to grant access.
3. Security manager controls the creation of class loaders.
4. Applications set security manager once.
5. Security manager first is defined and then implemented in code.

Conclusion

We have seen that there are two basic aspects to security in the Java programming environment. The first is security inside the JVM, and the second is security outside the JVM. In this article, we have concentrated on the first part of Java security, known as the "sandbox."

In the sandbox model, the bytecode verifier, classloader, and security manager all work in a cooperative manner to strengthen the security features of Java.

It has been said that the only really secure machine is one which is enclosed in a room with no entry capability, no network connectivity, and no console. Perhaps in time that requirement will be extended to no power also.

However, in the real world, where Java is clearly making its presence known, we can be assured that its developers had a strong security model in mind during its design.

As with all new technologies, there is a period of time during its evolution when weaknesses will be exposed, and there have been a few in Java's security model. The security features in the JDK1.2beta3 release is likely to provide interested parties sufficient evidence that Java has evolved to satisfy stringent security requirements.

The time has arrived when we can put away the crystal ball and make informed decisions regarding the use of Java in applications where security is an important consideration.

musings

I do not mean to bash engineers. I am part engineer myself and have often been employed by people needing an engineer interpreter. I found that I had a talent for listening to engineers, then explaining what they would say to me in terms that others, the mere mortals of the world, would understand. I even learned how to ask questions in an appropriately humble and self-deprecating fashion so the engineers would deem me worthy of their attention and time.

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>

We live in a world made marvelous by the doings of engineers, from the high tech of computers and cellphones to the subtle but all-important underpinnings of our society, such as sewers and water filtration systems. Without engineers, we would be living in the stone age. But until engineers learn how to design machines and software for mere mortals, their successes will be limited. What stands in the way today is an element of design known as the state machine.

Please keep in mind as I ramble that I consider software designers (programmers) as engineers, and this bit of prose will start making more sense and eventually get to the point.

A Different Path

As you might have gathered, I believe that engineers live in a slightly different world from the rest of humanity. Like a bit of science fiction, the world in which engineers live is in a slightly offset space, where everything can be expressed more precisely and every event has a totally predictable outcome. When the engineers interact with other people, their skewed worldview (from the perspective of others) can get in the way of smooth and empathetic communications. It also leads to many surprises for the engineers, because people are not nearly as predictable as physical models or programs.

For example, my first software engineer boss was a brilliant guy. He could bum lines of code out of anything I wrote. We were working on an embedded system, and we really interfaced well when it came to single-stepping through some gnarly assembler. But when it came to working with other people, well, it was interesting to say the least. I remember one time when I could hear him thinking aloud about giving the secretary (the one who worked for all the engineers) a makework task that would take her days. When he presented her with the task, he began (with no trace of sarcasm), "You won't mind doing this . . . ." I merely cringed. She later wept, then quit.

I am sure you have met or worked with people like this. These people are valuable, or we would have shot (or stoned) them long ago. And they can be quite fun to be around. I really enjoyed the engineering majors when I lived in a college dorm. The chemical engineers would describe exactly what was happening to them as they got drunk. The electrical engineers would stick a knife precisely into a toaster, then ask someone to hold it. The unfortunate volunteer would get a non-fatal shock because the engineer was using the heating coils inside the toaster to create a resistor bridge to reduce the voltage to a "safe" level. The crepe rubber soles of the engineer's shoes protected him.

Real Fun

While I am on the topic of household appliances, consider for a minute the humble, but much abused, VCR. Very likely, if you are a member of USENIX, you are somewhat of an engineer yourself. You can determine this by taking a quick test: do you find VCRs difficult to program? If you answered yes, you have very little engineering blood

The entire desktop design metaphor is a maze of states.

in you. If you wonder why anyone has trouble programming VCRs, please keep reading.

The secret to the programmable nature of VCRs is a simple microprocessor, quite commonly a Motorola 6805 descendent. By pressing a key on the remote control, you make a menu of choices appear. By pressing the correct sequence of keys, you can instruct the VCR to turn itself on, switch to channel 9, and begin recording at 10:00 pm on August 13 at the extended play speed. The remote control (or front panel) keypad is the input device, and the menus displayed on the TV screen are the output.

The VCR's microprocessor is executing a simple programming algorithm known as a state machine. Up to a point, each key pressed displays another menu, representing another state. Then the keys can be used to set parameters, such as starting time. When the user finds him- or herself in a state, pressing a particular key has a meaning pertinent only within this state. The programmer has made this so. The unfortunate user may not have made the deductive leap (never having implemented a state machine) that each button is not like a light switch, but can be used for many different purposes. It all depends upon what state the system is in.

This brings me back to my first programming boss. His triumph of human-machine engineering included a 16-button keypad, where most keys had four different purposes, depending upon the order in which you pressed the keys. Being a programmer and part engineer myself, I quickly caught on to this (yes, I find VCRs trivial to program, and digital watches, too). It wasn't until I was sent out on the road to teach customers about this incredibly simple user interface that I discovered how really inscrutable state machines could be.

After I quit that job and moved to California, I vowed that I would make computers easier to use. Not that I ever did – it was too tempting to create elegant designs instead.

More Than Human

Now let's consider the ultimate, the most popular computer-human interface yet developed: the desktop and the mouse. The mouse is a no brainer, right? It has one button that does whatever the programmer wants it to do, depending upon where the indicator (the cursor) is and when the user pushes the button. Instead of a 16-key input device, we have a single-key input device. But wait, there is that 101-key keyboard just sitting there in front of the user. By combining key presses, perhaps key combinations, and mouse clicks, we can "overload" the mouse button a wee bit more.

Of course, we can include state machines that make the designers of VCRs moan with envy. If we hold down the mouse button when the cursor is in the correct position, a menu appears. Then, if we drag the cursor to a position indicated by a triangle that represents an arrow, yet another menu appears, which might contain still more entries with triangles representing arrows. The perfect user interface: with a single button press and a few deft wrist movements, the user can navigate through several states – the ultimate in overloaded operators.

The entire desktop design metaphor is a maze of states. Move the cursor to the menu bar and click, and one thing happens. Move it inside one window, and clicking changes the cursor into a text entry cursor, and you can enter text. Move it to another window and click, and you can select objects or draw lines. In another window, you can use the mouse to play games. With three-button mice, clicking on the background window brings up different menus. The potential for new states is endless.

Babies

You might think I am daft to be complaining about the mouse/desktop design metaphor. Millions of people, tens of millions, have learned how to use a mouse and to be productive using computers, you say. Perhaps you have seen children, even toddlers, quickly figure out how to use a mouse. They move their hand and a funny mark moves around on the screen, it's natural, right? And the alternative, text-based, command line-oriented control, is only for real engineers. Frankly, I personally doubt that Windows would have ever had caught on without the training program known as Solitaire.

And what about the desktop metaphor itself? The original idea, from Xerox PARC and successfully promoted by Apple Computer, was to display an area with objects on it, called the desktop, which the user manipulated using the mouse. Users could drag things they didn't want into a trash can, drop something they wanted to print onto a printer, or double-click on something they wanted to "open" (another bit of programmer-speak). Notice that the desktop itself is free of state machines – or perhaps it can be considered a machine with one level of state. Still, the desktop avoids the many levels that cascading menus impose or changing to new state machines by simply moving the cursor into a different window.

Notice that the desktop itself is free of state machines – or perhaps it can be considered a machine with one level of state.

Now I don't know about your office desktop, but I use another programming metaphor on mine. It's called stacks. When I put something into a stack, it stays there until I need it. I use a FIFO for bills, a special, fast-moving stack for checks, a very deep stack for things that I should read but will never get around to. It works well for me as long as nobody attempts to clean up my desk. Then I can't find anything.

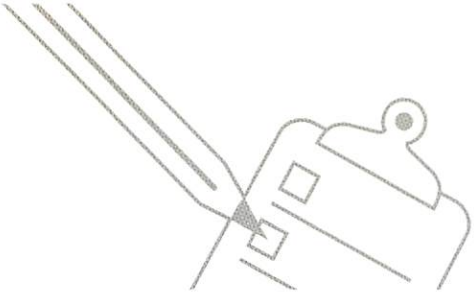
I would like to suggest humbly that the state machines that programmers and engineers are so fond of may not be the best human interface design ever conceived. The mouse works so well because we humans are accustomed to using our hands to move things. But designs that use a single button to do many different things, depending on the current state, do not so neatly correspond to the outer world, where a light switch only turns on or off a light and a rock is only a rock.

Oops, there are fader light switches. And humans have used rocks for grinding grain and smashing enemies. But I don't know if a rock user would ever have figured out that if he presses down on the rock, slides it up to the arrow, then to the right, down to another arrow, right, and then down again, and finally releases, that something different, but desirable, will happen. State machines just aren't natural.

Virtual Reality

I don't have a solution to this quandary. If I had one, I would have to keep it a secret until I could patent it and become the richest man in the world (just kidding). But I do think that an answer lies nearby, by examining the real-world behavior of our fellow humans (and engineers) and applying what we learn to the human-computer interface.

The spatial metaphor of the desktop works well, as does the mouse-moving-cursor metaphor. Perhaps the interface of the future will require 3D, with tiny displays "painted" on our retinas simulating an environment with depth, and input devices better designed to mimic our manipulation of "real"-world objects. I don't think the answer involves putting more buttons on the mouse. But I do invite you to ponder this. Perhaps you will become the richest person in the world, or maybe just the most generous.



standards reports

by Nicholas M. Stoughton

USENIX Standards Liaison

<nick@usenix.org>

Cooperative Development – A Simple Model

Lowell Johnson
<lowell.johnson@unisys.com>

The Portable Applications Standards Committee (PASC) of the IEEE Computer Society has been debating the future of POSIX for some months.

It is clear that many of those in the standards industry are extremely disturbed by the overlapping development effort that goes on between PASC and The Open Group (TOG). There is POSIX.1 and POSIX.2, and there is the Single UNIX Specification (SUS). SUS is a proper superset of POSIX; everything in the POSIX standards that it is based on is in SUS, and in the case where there appears to be conflict between the two standards, POSIX wins. However, implementers have to prove conformance to two standards and have to spend effort developing two standards.

The current objective of PASC is to move to a collaborative method of working whereby only one standard is produced: a single document, written by all the interested parties, and adopted in all the groups that wish to have it as a standard. This article proposes a method for cooperative development to produce such a single standard.

The simple model proposed here is exactly that: simple. Even though a few changes may be needed and details need to be worked out, this is the best chance for all parties involved in POSIX development and maintenance to work in a cooperative environment. The groups initially targeted at this work are: The Open Group, ISO/IEC JTC1/SC22/WG15, and the IEEE PASC group itself. However, the model is easily extendable to additional groups if needed.

The diagram below shows the basic steps, which are:

1. Writing or revising the standard
2. Balloting the draft
3. Remediation of possible objections

Write	Ballot	Review	Ballot
All Interested Parties	Individual (IEEE Rules) Entity/Corporation (TOG Rules) National Body (ISO Rules)	All Interested Parties	Individual (IEEE Rules) Entity/Corporation (TOG Rules) National Body (ISO Rules)

Writing or Revising the Standard

This is logically the simplest part of the process, but the practical logistics may be cumbersome. Basically, the work is done at an announced time and place and is open to anyone to participate for a small fee (approximately the current PASC fees). Anyone may chair these meetings, but the logical choices would be either the PASC or TOG leaders in the technical area.

The frequency of meetings should also be fairly straightforward. Projects undergoing active development need to meet six to eight times a year (or more) to facilitate the faster turnaround times the industry needs and expects. These are working group meetings, dealing mainly with issue resolution. Some of these meetings should probably coincide with the quarterly PASC meetings and quarterly TOG meetings, though this is not a necessity.

Groups not currently working on active development would meet once or twice a year in conjunction with either TOG or PASC. Since WG15 has approved a plan to meet only once a year at either a PASC meeting (included in the above considerations) or the annual SC22 meeting (not appropriate for a work project meeting), no special consideration for meetings with WG15 should be necessary.

The only contentious issue in this phase is the process for making decisions needed before formal balloting begins. TOG and WG15 are used to formal voting while PASC employs a looser consensus process (at this stage). The compromise is simple: attempt a consensus solution, but failing that, take a majority vote. Decisions which are of major importance may be requested to be postponed until the next meeting if it is considered that a significant number of the major participants are not in attendance (or even better, a vote may be taken electronically after the meeting).

For example: in a contentious issue about whether or not to include XYZ, it is noticed that 2 of the 3 vendors of XYZ are absent. There is obviously no consensus, so a vote is deferred until the next meeting. Eventually over 50% approve XYZ, so it is included in the draft (note that in the balloting phase over 75% must approve the inclusion in IEEE).

The most difficult aspect initially will be to get the PASC and TOG logistical folks to coordinate the meetings as appropriate. Since these are now done independently and up to a year in advance, modifying our current schedules will be difficult. Once we get over the first year or so, the future meetings should fall into place naturally.

Balloting the Draft

The model here is simple to describe, but a bit harder to coordinate. Simply stated, each group ballots using its own method: IEEE votes by individual, TOG by corpo-

ration, and WG15 by country. The difficulty comes in coordinating the balloting. The IEEE and TOG ballot periods are somewhat similar, and their timeframes could be adjusted a bit if necessary. The difficulty is the ISO process, both because there are more defined ballot points and because the ballots take longer. However, most ISO comments are either editorial or copies of ballots already submitted in one of the other ballots, so resolving the comments is not the major problem. We already have a working synchronization plan between PASC and WG15, so we have evidence that it is possible!

A whole new synchronization plan probably needs to be developed, but let's start with the following model.

1. Information and Preliminary ballots are done with drafts before the draft that goes to formal IEEE ballot and formal company review in TOG.
2. CD ballot (and equivalent) begins with the first IEEE ballot.
3. Final CD ballot is the draft approved by the corporations and IEEE. This should not be a problem since the final CD ballot under the new rules does not allow substantiative changes anyway, so there should not have to be any additional remediation with IEEE or the ToG corporations for these changes.

Another problem is what to do when formal balloting begins, and one group approves it and the other does not. There is no requirement that all groups approve a particular document, but it would be desirable. To that end I propose the following compromise.

When a document reaches the stage where either IEEE or TOG approves it (but not both), one final remediation process is performed. The revised document is then recirculated. If both groups approve, we are done. If the approving group still approves, and the disapproving group still disapproves, we are also done, but in this case only the one group adopts the standard and the other group drops the work item, and does not pursue it independently.

The last point is important and must be made a condition of participating in a joint effort such as this. Otherwise, the whole concept of a single standard developed by a single group is worthless. Each group much agree at the beginning of the development process to abide by this rule.

ISO was not mentioned in this last issue because either IEEE or TOG could (or should) advance the work to ISO.

Remediation

Resolving comments and objections after a ballot should be done by the whole group in an open process similar to the development phase. However, what normally happens is that many of the original developers drop out at this point for one reason or another (it is a bit tedious after all). The only requirement this model would impose would be to ensure that all the participating groups are still represented by active participants during this phase.

Editor's Note: Lowell Johnson is Chair of the IEEE Portable Applications Standards Committee, PASC. This article is a personal submission from him into the debate, and cannot be taken as a statement of PASC policy. This proposal is under discussion and deliberation by members of all three groups. It will be debated during the July PASC meeting in Nashua, New Hampshire. Watch this space for further news.

POSIX Realtime Extensions Subgroup

Joe Gwinn <j.gwinn@ed.ray.com> reports on the April 1998 meeting in Dallas, TX.

POSIX.13, after years of effort, is now IEEE Std 1003.13-1998, having been approved by the IEEE Standards Board on 19 March 1998.

The System Services Realtime subgroup (SSWG-RT) has three other approved standards: POSIX.1b-1993 (realtime extensions, was called POSIX.4), POSIX.1c-1995 (pthreads, was called POSIX.4a), and POSIX.1i-1995 (technical corrections to POSIX.1b). These base standards and their corrigenda have all been incorporated into the published standard ISO/IEC 9945-1:1996, as discussed in greater detail below. IEEE Std 1003.13-1998 (realtime profiles) will remain a freestanding document.

SSWG-RT has four active formal projects: P1003.1d (more realtime extensions, was called P1003.4b), P1003.1j (yet more realtime extensions, was unofficially called P1003.4d), P1003.1q (Trace), discussed below, and POSIX.1n (technical corrections to 1003.1c).

POSIX.13 is a family of four related realtime profiles ranging in size from the very small through to a full-featured platform conforming to essentially all of POSIX.1-1990, POSIX.1b-1993 and POSIX.1c (threads). This set of profiles is also known as the "slice and dice" profile set, since it allows subsetting of the base standard. Such subsetting, allowing mandatory features of the base standard to be omitted, is not normally allowed in POSIX profiles, and POSIX.13 is a special case.

The smaller profiles specify just that subset of POSIX interfaces needed to "clothe" widely used small kernels such as pSOS (from ISI), VxWorks (from Wind River), MTOS (from IPI), VRTX32 (now from Mentor, originally from Hunter&Ready), and the ORKID interface standard (from VITA), which although very similar in approach and function, differ greatly in interface details. As a matter of interest, there are more of these small kernels in UNIX systems than there are UNIX kernels because, for instance, many I/O controllers and peripherals themselves use one of these small kernels.

Standardization of these interfaces will yield the same benefits for embedded and realtime systems as standardization of

UNIX did for workstations. In addition, the POSIX.13 interfaces are chosen to allow multi-computer distributed systems to be built, such as those used in factory automation. Such systems are typically set up as a hierarchy, with a few large-profile machines at the top, and a large number of smaller profile machines at the bottom controlling this or that piece of machinery, perhaps with an intermediate layer of supervisory machines between top and base, and all communicating with peers, superiors, and subordinates, as needed.

Other work within SSWG-RT includes POSIX.1d, where Draft 10 went out for its first recirculation in March 1997 (with Interrupt Control, a particularly contentious area, and Device Control, deferred and moved into non-normative annexes). The last ballot closed on 21 April 1997, at 45% affirmative (30% short of the required 75%), with a low fraction of the ballot group voting or responding to ballot resolution emails. The Ballot Group, which was quite stale, has therefore been reformed.

POSIX.1d contains a number of realtime interfaces and options that arrived too late to be included in POSIX.1b. The major new interfaces and options are: `spawn()`, a functional merger of `fork()` and `exec()`, needed both for efficiency and to allow use on platforms lacking memory management hardware; a sporadic-server scheduling policy, used to prevent asynchronous high-priority processing from totally consuming the computer; cpu-time clocks and timers, used to both measure and bound use of cpu by processes; `devctl()`, the successor to the `ioctl()` of classic UNIX; Interrupt Control, a set of interfaces intended to allow direct application-level control of devices such as array processors and radar signal processors; and Advisory Information, a set of interfaces that allow an application to declare to the kernel that for instance a specified file will be read sequentially, allowing the kernel to optimize performance. A number of existing interfaces are also being augmented by the addition of variants supporting timeouts.

POSIX.1j (advanced realtime) went out for its first recirculation in January 1998. It achieved only 36% approval at this point.

POSIX.1j contains a number of realtime interfaces and options that arrived too late to be included in POSIX.1d. The major new interfaces and options are: Typed Memory, a set of interfaces supporting the `mmap()`-like mapping of diverse kinds of physical memory (e.g., SRAM, DRAM, ROM, EPROM, EEPROM) via multiple and/or diverse physical paths used for instance to access special hardware and memory via attached VME busses; `nanosleep_abs()`, a high-resolution `sleep()` allowing the user to specify when to awaken, rather than how long to sleep; Barrier Synchronization, a set of interfaces intended to support efficient implementation of parallel DO/FOR loops on massively parallel computers; Reader/Writer Locks, used to allow efficient parallel access to data in situations where reads vastly outnumber writes; Spinlocks, a very fast synchronization primitive for use on shared-memory multiprocessors; and Persistent Notification for Message Queues, an option for 1003.1b-1993 Message Queues.

POSIX.1q proposes trace interfaces that have been the focus of considerable interest and much discussion over the past two years, with joint meetings of SSWG-RT and SSWG-SRASS being held. The Trace API Small Group, composed of people from both SSWG-RT and SSWG-SRASS, with significant representation from interested vendors, has developed a complete set of APIs and corresponding rationale, resulting in the draft standard currently under intensive development in the Working Group. The purpose of these trace interfaces is to allow the collection and presentation of trace logs of application calls on the operating system, I/O activity, user-defined events, and the like, with an eye to debugging user code running at essentially full speed. Hardware and kernel faults may also be recorded; the Trace APIs are general. Tracing is a requirement for realtime systems. A number of vendors have come forward with implementations, and there has been great interest in coming up with standard APIs and, to a lesser extent, log file contents. The proposed trace interfaces do not in the least resemble standard inspect-and-change debuggers and require no kernel knowledge or access to use.

About the Author

Joe Gwinn is vice chair of the System Services Working Group; chair of the SSWG Realtime Extensions Subgroup, 1003.4 Atavar; and Defender of the Realtime Faith.

ISO C Amendment 1 (MSE)

David Lindner and Finnbarr Murphy

The Single UNIX Specification, Version 2 includes in its System Interfaces Specification (XSH) the ISO/IEC 9899:1990/Amendment 1:1995 (E) to ISO/IEC 9899:1990, Programming Languages – C (ISO C). This paper is a brief introduction to this extension. It is assumed that the reader is familiar with the C language and has some basic understanding of internationalization concepts and character encoding methods.

Introduction

ISO C Amendment 1 (MSE) was part of the first amendment made to the ISO C standard. The MSE consists of a set of library functions that provide a relatively complete and consistent set of functions for application programming using multibyte and wide characters.

The other major items included in this amendment are digraphs, alternate

spellings for several C tokens, and the header <iso646.h>. These items are not discussed here since they are outside the scope of this paper.

The ISO C standard laid some groundwork for multibyte and wide character programming by providing a small number of multibyte and wide character functions. The working group decided to wait for the C developer community to acquire more experience with implementing multibyte and wide character libraries before extending this model further.

A working group (ISO/JTC1/SC22/WG14) was set up to study the various existing implementations and developed the Multibyte Support Extension as part of the first amendment (called C Integrity) to the ISO C standard.

The System Interfaces Specification, XSH, Issue 4, Version 2, which was developed in 1994, incorporated a draft version of the MSE. XSH, Issue 5 incorporates the final version of the MSE.

Extended Characters

We traditionally think of characters as one byte entities represented by the **char** data type. This is simple, but allows for a maximum of 256 distinct characters.

In the MSE model, the concept of a character has been extended. Extended characters can be represented in three ways:

- multibyte character encodings
- wide character encodings
- generalized multibyte encodings.

A multibyte character is a sequence of one or more bytes that can be represented as an array of type **char**; in other words, a single character may occupy one or more consecutive bytes. An example of such an encoding is EUC (Extended UNIX Code). EUC provides a structure by which any number of codesets may be encoded into a multibyte encoding.

The primary advantage to the one byte/one character model is that it is very easy to process data in fixed-width chunks. For this reason, the concept of the wide character was invented. A wide character is an abstract data type large enough to contain the largest character that is supported on a particular platform. To date, most system implementors have chosen 32 bits, although there are implementations with 16-bit and 8-bit wide characters. It should be noted that although many vendors have chosen a 32-bit wide character, because the wide character is an abstract type, it is not guaranteed to be the same across all platforms.

To support the concept of wide characters, the MSE defines the integral type **wchar_t**. However, it does not define the size of **wchar_t**, but states it shall be as wide as necessary to hold the largest character in the code sets of the locales that an implementation supports.

In addition to the traditional concept of the multibyte character, the MSE has added the concept of the generalized multibyte character.

Multibyte Characters

There are many different multibyte encoding schemes, but these can be broken down into three basic categories:

- restartable multibyte encodings
- stateful multibyte encodings
- generalized multibyte encodings.

Restartable multibyte encodings are defined such that if you were to process a multibyte data stream, it would be possible to determine the correct separation of characters no matter where you were positioned in the data stream. In the case of stateful encodings, you need one extra piece of information to be able to correctly process characters in the data stream. This extra piece of information is commonly referred to as the state of the data stream.

Why must we be able to unambiguously restart a data stream? If any byte sequence can have more than one meaning as a sequence of characters, then the multibyte code is ambiguous; that is, you could have multiple meanings for the same data stream depending upon where you started in the data stream. For example, the following multibyte encoding is not restartable:

```
0x41 0x42 0x61 0x62 0x43
```

In this particular encoding, the combination of 0x61 and 0x62 produces an "F." If we start processing this string at the beginning, all the characters would be processed correctly and the result would be the string:

```
A B F C
```

If we start processing the string at 0x62, then the result would be the partial string:

```
b C
```

In a restartable encoding, the conversion interfaces would have recognized the 0x62 as an illegal multibyte character, and our program could choose to ignore that illegal character and move on, or perhaps it might try to back up and see if it could form a complete multibyte character.

In restartable multibyte encodings, each byte sequence in a particular encoding scheme stands for one character; the same character regardless of context. Stateful multibyte encoding schemes have a concept of shift state; certain codes called shift sequences effectively change the data stream to a different shift state, and the meaning of byte sequences is changed according to the current shift state.

If we use the same multibyte encoding and make it a stateful encoding, we will introduce two new operators called shift state operators, SS0 and SS1. The default shift state for this particular codeset is SS0. In this example, the 0x61 in its

shifted state produces an "F," and in its default state produces an "a":

```
0x41 0x42 SS1 0x61 SS0 0x43
  0x61
```

Since the default shift state is SS0, the above sequence of bytes should produce the string:

```
A B F C a
```

The stateful multibyte encodings are not restartable either, because if we started processing the string after a shift state operator, we could potentially get the wrong string.

Normally, if you try to pass a string containing multibyte characters to a function that does not know about them, such a function treats a string as a sequence of bytes, and interprets certain byte values specially; for example, the null byte, the slash character. Since it is illegal for a multibyte character to use any of the special byte values as part of its encoding, the function should pass it through as if it were a single byte string. (Note: The multibyte encoding may still use the slash or null byte, it just cannot use them as part of another multibyte character.)

This is where the concept of the generalized multibyte encoding arises. Traditionally, we think of multibyte encodings as file code and wide characters as process code, where file code resides on disk and process code is used by an application. This is not to say that multibyte encodings are not used by applications. Indeed many applications today use multibyte encodings routinely, but because they do not require the ability to process characters as discreet chunks they have no need to convert the multibyte encodings to wide characters.

In summary, generalized multibyte encodings can be encoded in any way. The special byte values discussed above have no meaning in generalized multibyte encodings. Functions that have no concept of multibyte encodings would fail if they tried to process generalized multibyte encodings. By defining the concept of generalized multibyte encodings, we provide a method by which we can say a particular file is associated with a particular locale, and can only be processed by specific routines running in this locale. Generalized multibyte encodings are more of a logical grouping than a specific definition. They provide us with a way to associate files with specific locales and codesets, and allow us to safely operate on those files as long as we are in the proper locale. The important restriction is that generalized multibyte characters can never be processed directly, they can exist only on disk. (Note: Processed refers to the parsing routines available in C. Any file may be processed as binary data.)

To take an example of a generalized multibyte encoding, Unicode is a 16-bit codeset that can be found on Windows 95 and Windows NT. One of the problems with Unicode is that it has NULL bytes embedded in its encoding. For example, the string:

```
a b c
```

is actually encoded as follows:

```
0x00 0x61 0x00 0x62 0x00 0x63
  0x00 0x00
```

Those who are familiar with any of the string handling routines in C, can see that these routines will have problems with this string. Similarly, if you tried to read this file from a disk as a text file you would have problems. However, with the concept of generalized multibyte encodings we can say this file is associated with a Unicode locale, and the stdio routines can be smart enough to know that when they are in the Unicode locale they can read the Unicode file properly.

Headers

The MSE defines two headers to support the new functionality:

- `<wctype.h>`

  Contains the declarations for the functions analogous to those in `<ctype.h>`; that is, the classification and mapping functions.

- `<wchar.h>`

  Contains the remaining declarations, including the following types:

  - **`wchar_t`**

    An integer type whose range is large enough to represent all distinct values in any extended character set in the supported locales. Known as the wide character type.

  - **`mbstate_t`**

    Stores the current parse state of a stream.

  - **`wint_t`**

    An integer type that can hold any wide character and WEOF.

Character Classification and Mapping Functions

Character classification determines whether a particular character code refers to an upper-case alphabetic, lower-case alphabetic, alphanumeric, digit, punctuation, control or space character, or any one of a number of other groupings.

Mapping functions are sometimes called case conversion functions, because the original mapping functions simply mapped upper-case to lower-case and vice versa.

In the past, macros were often used to classify or map character codes. This was possible since the assumption was that an application was dealing with ASCII characters. Today, classification functions are used which classify wide character codes according to the type rules defined by the category LC_CTYPE of the application's current locale.

In the ISO C standard the behavior of character classification functions is affected by the current locale. Some functions have implementation-dependent behav-

ior when not in the POSIX locale. For example, in the POSIX locale, `isupper()` returns true (non-zero) only for uppercase letters. The MSE contains no description of how the POSIX locale affects the behavior of the above functions, but states that when a character c causes an `isxxx(c)` function to return true, the corresponding wide character wc shall cause the corresponding wide character function to return true. Note, however, that the converse is not true.

The ISO C standard defines 11 classification (also known as character testing) functions. The MSE defines an analogous set of wide character classification functions, returning non-zero for true and zero for false, for example `iswalnum()` is analogous to `isalnum()`.

As the number of defined locales increased, the requirement for additional character classes increased. For example, while a classification function such as `isupper()` makes perfect sense in the English language, it does not make any sense in a language such as Japanese that has no concept of case. Conversely, a function such as `iskana()` makes perfect sense for Japanese, but doesn't make any sense in English. For this reason, the MSE defined a number of extensible wide character classification functions – `wctype()`, `iswctype()`, `wctrans()`, and `towctrans()` – as general-purpose solutions to this problem.

These two functions are generally used in combination. However, sometimes the `wctype()` function is used on its own by an application to test whether a character classification is available in a specific locale. If the current setting of the LC_CTYPE locale changes between calls, the behavior is undefined.

The MSE specifies that the following code segments are equivalent to each other:

```
iswctype(wc, wctype("alnum"))      iswalnum(wc)
iswctype(wc, wctype("alpha"))      iswalpha(wc)
iswctype(wc, wctype("cntrl"))      iswcntrl(wc)
iswctype(wc, wctype("digit"))      iswdigit(wc)
iswctype(wc, wctype("graph"))      iswgraph(wc)
iswctype(wc, wctype("lower"))      iswlower(wc)
iswctype(wc, wctype("print"))      iswprint(wc)
iswctype(wc, wctype("punct"))      iswpunct(wc)
iswctype(wc, wctype("space"))      iswspace(wc)
iswctype(wc, wctype("upper"))      iswupper(wc)
iswctype(wc, wctype("xdigit"))     iswxdigit(wc)
towctrans(wc, wctrans("tolower"))  towlower(wc)
towctrans(wc, wctrans("toupper"))  towupper(wc)
```

Number Conversion Functions

Three new functions are included to facilitate conversion from wide character strings (also known as wide strings) to a variety of numeric formats. These are the wide character versions of the ISO C functions `strtod()`, `strtol()`, and `strtoul()`.

These functions work as follows:

- First, the function decomposes the wide character string into three parts:
  - An initial, possibly empty, sequence of white-space wide characters as determined by the `iswspace()` function
  - a subject sequence interpreted as either a floating point constant, **long** or **unsigned long**
  - a final sequence of one or more unrecognized wide character codes including the terminating null wide character.
- The function then attempts to convert the subject sequence into the required number format by parsing the subject sequence and returning the result. If the subject sequence is empty or does not have the expected form, no conversion is performed.

In other than the POSIX locale, implementation-dependent forms of a subject sequence may be supported.

String Handling

Sixteen new wide character string functions are defined. Most are similar to their char-based counterparts. For example, `wcscopy()` is analogous to `strcpy()`, but operates on wide strings. In general, the data types of some parameters differ, but the purpose of the parameters is the same.The comparison functions `wcscmp()` and `wcsncmp()` compare two wide character strings by comparing the wide characters based on the character's encoded value, while the `wcscoll()` function compares each wide character interpreted according to the collating sequence information specified by the LC_COLLATE category of the current locale.

The `wcsxfrm()` function transforms a wide character string and places the result in an array of wide characters. The transformation is such that if the `wcscmp()` function is applied to two transformed wide character strings, the result is the same as if the two wide character strings were compared using `wcscoll()`. Both wide character strings must be transformed using `wcsxfrm()`. It is invalid to compare a transformed string to a non-transformed string. Note that no function is defined to restore a transformed string to its original layout.

When wide character strings are likely to be compared more than once, it is more efficient to transform them using `wcsxfrm()`, compare them using `wcscmp()`, and retain the transformed strings for subsequent comparisons.

The MSE also defines a number of wide character array functions. These functions operate on arrays of type **`wchar_t`** whose size is specified by a separate count argument. These functions are not affected by locale and all **`wchar_t`** values are treated identically, including the null wide character and wide characters not

corresponding to valid multibyte characters. Thus, the `wmemcmp()` function compares each wide character array element using the encoded value of each wide character.

The Input/Output Model

The MSE input/output model assumes that characters are handled as wide characters within an application and stored as multibyte characters in files, and that all the wide character input/output functions begin executing with the stream positioned at the boundary between two multibyte characters.

The definition of a stream was changed to include the concept of an orientation for both text and binary streams. After a stream is associated with a file, but before any operations are performed on the stream, the stream is without orientation. If a wide character input or output function is applied to a stream without orientation, the stream becomes wide-oriented. Likewise, if a byte input or output operation is applied to a stream with orientation, the stream becomes byte-oriented. A new function `fwide()` is used to determine or alter the orientation of a stream.

Byte input/output functions cannot be applied to a wide-oriented stream and wide character input/output functions cannot be applied to a byte-oriented stream.

While wide-oriented streams are sequences of wide characters, the external file associated with a wide-oriented stream may be an implementation-dependent multibyte encoding. Furthermore, it is acceptable that the file associated with this stream is a generalized multibyte encoding such as Unicode.

Note that the input/output model does not preclude applications from storing data in external files as wide characters.

Conversion Functions

As discussed earlier, multibyte character streams may have state-dependent encodings. To handle state-dependent encodings, the MSE includes the concept of a conversion state that is associated with each FILE object that affects the behavior of a conversion between multibyte and a wide character encoding.

The conversion state information augments the FILE object's information about the current position of the multibyte character stream with information about the parse state for the next multibyte character to be obtained from the stream. For state-dependent encodings, the remembered shift state is part of this parse state. Every wide character input or output function makes use of this state information and updates its corresponding FILE object's conversion state accordingly.

The non-array type **`mbstate_t`** is defined to encode the conversion state under the rules of the current locale and provide a character accumulator. This implies that encoding rule information is part of the conversion state. No initialization function is provided to initialize **`mbstate_t`**. A zero-valued **`mbstate_t`** is assumed to describe the initial conversion state. Such a zero-valued **`mbstate_t`** object is said to be unbound. Once a multibyte or wide character conversion function is called with the **`mbstate_t`** object as an argument, the object becomes bound and holds the conversion state information which it obtains from the LC_CTYPE category of the current locale. No comparison function is specified for comparing two **`mbstate_t`** objects.

The MSE assumes that only wide character input/output functions can maintain consistency between a stream and its corresponding conversion state. Byte input/output functions do not manipulate or use conversion state information. Wide-character input/output functions are assumed to begin processing a stream at the boundary between two multibyte characters. Seek operations reset the conversion state corresponding to the new file position.

The `mbsinit()` function is specified because many conversion functions treat the initial shift state as a special case and need a portable means of determining whether an **`mbstate_t`** object is at initial conversion state.

The MSE provides a method to distinguish between an invalid sequence of bytes in a multibyte stream and a valid prefix to a still incomplete multibyte character. Upon encountering such an incomplete multibyte sequence, the functions `mbrlen()` and `mbrtowc()` return `-2` instead of `-1`, and the character accumulator in the **`mbstate_t`** object may store the partial character information. This allows applications to convert streams one byte at a time or even to suspend and resume conversion if required. The conversion functions are thus said to be restartable.

The function `btowc()` is used to determine whether its argument is a valid multibyte character in the initial shift state, and to return the corresponding wide character. The function returns WEOF if the character has a value of EOF or if it is not a valid multibyte character in the initial shift state.

Similarly, the function `wctob()` is used to determine whether its argument is a member of the extended character set whose multibyte character representation is a single byte when in the initial shift state, and to return the corresponding single byte character. The function returns `EOF` if the character does not correspond to a valid multibyte character of length 1 in the initial shift state.

The MSE specifies a number of restartable functions which take as their last argument a pointer to an object of type **`mbstate_t`**. If the pointer is NULL, each function uses its own internal **`mbstate_t`** object instead, which is ini-

tialized at startup to the initial conversion state. Note that, unlike their corresponding ISO C standard functions, a function's return value does not represent whether the encoding is state-dependent. These functions are:

MSE	ISO C	Purpose
mbrlen( )	mblen( )	Determine the length in bytes of a multibyte character.
mbrtowc( )	mbtowc( )	Convert a multibyte character into a wide character.
wcrtomb( )	wctomb( )	Convert a wide character into a multibyte character.
mbsrtowcs( )	mbstowcs( )	Convert a multibyte string into a wide character string.
wcsrtombs( )	wcstombs( )	Convert a wide character string into a multibyte string.

A more detailed explanation of two of the above functions will help to clarify the concept of restartable functions.

The function `mbrtowc()` inspects at most n bytes to determine the number of bytes needed to complete the next multibyte character. If a multibyte character can be completed, `mbrtowc()` determines the corresponding wide character and returns it in `*pwc`. If the corresponding wide character is the null wide character, the conversion state is reset to the initial conversion state. This function returns one of the following:

`(size_t)-2`
: The next n bytes contribute to, but do not complete, a valid multibyte character, and all n bytes have been processed.

`(size_t)-1`
: An encoding error has occurred. The next n or fewer bytes do not contribute to a valid multibyte character `errno` is set to EILSEQ. The conversion state is undefined. Note: `(size_t)-2` and `(size_t)-1` should be tested before the `>0` case.

0
: If the next n or fewer bytes complete a valid multibyte character that corresponds to the null wide character.

`>0`
: The number of bytes used to complete a valid multibyte character.

The function `mbsrtowcs()` is a restartable string conversion routine which converts a sequence of multibyte characters, beginning with the conversion state described by the **`mbstate_t`** object pointed to by `ps`, from the array indirectly pointed to by `src` into a sequence of corresponding wide characters pointed to by dst. Conversion continues up to and including a terminating null character which is also stored in `dst`. Each conversion takes place as if by a call to the `mbrtowc()` function. If an error occurs, errno is set to the macro EILSEQ and `mbsrtowcs()` returns `(size_t)-1`.

Conversion stops when one of the following occurs:

- A null multibyte character is encountered and processed. In this case the conversion state is reset to the initial conversion state and `scr` is assigned a null pointer.
- A sequence of bytes is encountered which do not form a valid multibyte character.
- When `len` multibyte characters have been processed.

Miscellaneous Functions

The `wcsftime()` function behaves as if the character string generated by the `strftime()` function is passed to the `mbstowcs()` function as the character string parameter, and the `mbstowcs()` function places the result in the `wcs` parameter of `wcsftime()`, up to the limit of the number of wide characters specified by `maxsize`.

This function uses the local time zone information. The format parameter is a wide character string consisting of a sequence of wide character format codes that specify the format of the date and time to be written to `wcs`.

More Information

More information on the Single UNIX Specification, Version 2 can be obtained from the following sources:

- The Open Group Source Book, *Go Solo 2 – The Authorized Guide to Version 2 of the Single UNIX Specification*", 500 pp., ISBN 0-13-575689-8. This book provides complete information on what's new in Version 2 , with technical papers written by members of the working groups that developed the specifications, and a CD-ROM containing the complete 3,000 page specification in both HTML and PDF formats (including PDF reader software). For more information on the book, see <http://www.UNIX-systems.org/gosolo2>.
- The Single UNIX Specification can be browsed and searched online at The Open Group WWW site, <http://www.UNIX-systems.org/go/unix>.

About the Authors

David Lindner is a principal engineer with Digital Equipment Corporation and a former member of The Open Group Internationalization Technical Working Group.

Finnbarr P. Murphy is a principal software engineer with Digital Equipment Corporation and is vice-chair of The Open Group Base Technical Working Group.

the bookworm

Books reviewed in this column:

John D. Blair
Samba
Seattle, WA: SSC, 1998. ISBN 1-57831-006-7. Pp. 298+CD-ROM.

Charles E. Perkins
Mobile IP
Reading, MA: Addison-Wesley, 1998. ISBN 0-201-63469-4. Pp. 275.

Vicki Brown & Chris Nandor
MacPerl: Power and Ease
Sunnyvale, CA: Prime Time Freeware, 1998. ISBN 1-881957-32-2. Pp. 372+CD-ROM.

Joseph N. Hall & Randal L. Schwartz
Effective Perl Programming
Reading, MA: Addison-Wesley, 1998. ISBN 0-201-41975-0. Pp. 273.

Ellie Quigley
The Complete Perl Training Course
Upper Saddle River, NJ: Prentice Hall, 1998. ISBN 0-13-079980-7. (book + 2 CD-ROMs)

Jan L. Harrington
SQL Clearly Explained
Chestnut Hill, MA: AP Professional, 1998. ISBN 0-12-326426-X. Pp. 279.

Mark F. Komarinski & Cary Collett
Linux System Administration Handbook
Upper Saddle River, NJ: Prentice Hall, 1998. ISBN 0-13-680596-5. Pp. 384+CD-ROM

Michael K. Johnson & Erik W. Troan
Linux Application Development
Reading, MA: Addison Wesley, 1998. ISBN 0-201-30821-5. Pp. 538.

Jayant Kadambi, et al.
Gigabit Ethernet
Upper Saddle River, NJ: Prentice Hall, 1998. ISBN 0-13-913286-4. Pp. 365.

Hattie Schroeder & Mike Doyle
Interactive Web Applications with Tcl/Tk
Chestnut Hill, MA: AP Professional, 1998. ISBN 0-12-221540-0. Pp. 603+CD-ROM.

(Continued on page 71)

by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

By the time this appears, Windows 98 will be available, and the publishers have been getting ready for the users: as of the end of May, I had received notices of 66 books on Win98.

No, you needn't worry. I don't use it. And I tell the publishers not to send them to me. Anyway, as far as I can tell, all one needs if one's employer insists on Win98 or NT is the books from O'Reilly (or a different job). Alternatively, you could get *Samba*, John Blair's book on a program that allows integration of real systems (e.g., UNIX) into a Windows network. Blair's book is a combination of tutorial, reference, and CD-ROM. It will be extremely useful for all of you involved with heterogeneous systems.

Wireless

Charles Perkins was on the program committee for the first USENIX Symposium on Mobile and Location-Independent Computing (August 1993). He has also, both while at IBM and now at Sun, been very active in the IETF's activities. (Among other things, he's one of the authors of the IAB draft "The Case for IPv6.") His *Mobile IP* explains everything I ever wanted to know about the protocols (including both IPv6 and DHCP [=Dynamic Host Configuration Protocol; RFC 2131]). It has both a glossary and a first-rate list of references. Perkins even manages to get in a paragraph on the home address option (draft-degermark-ipv6-hc-03.txt). The section on route optimization is also very fine. If you are at all interested in computing away from your desktop, you need this book.

Perl

There are three disparate Perl products on my desk as I write: Ellie Quigley's *The Complete Perl Training Course*, Hall and Schwartz's *Effective Perl Programming*, and Brown and Nandor's *MacPerl*. Except for the compulsive reviewer, I can't imagine these being read by the same audience.

MacPerl is a neat and elegant version of Perl5 for the Macintosh. The book and CD make an excellent package for those who will be supporting AppleScripts or AppleTalk (or other Macintosh applications) and want to use Perl. The book is terse, in the UNIX manual tradition, but not difficult. I liked it a lot. (This is the same group that brought us MkLinux last year – they're really on a roll!)

Effective Perl Programming is subtitled "Writing Better Programs with Perl," and I think the authors' point is made: with an understanding of this small volume, Perl programs will be clearer, more efficient, and, with luck, more elegant.

Quigley's "Training Course" is comprised of a Perl training course on CD-ROM, *Perl by Example* (2nd Ed.), and a second CD-ROM containing Perl distributions for Win95/WinNT/UNIX. It may be a great thing for the poor guy who's just been told that the company is using Perl starting next Monday.

Queries

If you use a database, you use SQL. If you have new staff that must understand what's happened when they query for a phone number or whether an invoice was paid or . . . , *SQL Clearly Explained* will be worth the investment. Jan Harrington really does explain things clearly, and her book should enable the beginner to comprehend things like "joins" and why using "difference" is equivalent to a negative.

Linux

Two really fine books have appeared on Linux. Komarinski and Collett have written a volume, *Linux System Administration Handbook*, that is the equivalent of Evi Nemeth et al. and Frisch's: a reliable sysadmin volume for Linux. It has a chapter on networking and a brief one on Samba. If you really need Samba, you're better off with Blair.

Johnson and Troan have produced *Linux Application Development*, which makes no concessions to the unwashed: it is designed for programmers and developers who are developing or porting Linux applications. Both authors are with RedHat, and their development experience makes this a superb piece of work.

Ethernet and Tcl

Stringing T1 lines around neighborhoods, or ISDN, or fiber is really neat. But what about "internal" bandwidth? Kadambi and his colleagues have the answer in *Gigabit Ethernet*. In brief, this is a clause-by-clause guide to much of IEEE 802.3z. And quite a good one.

A year ago, Brian Kernighan said that Tcl/Tk was the "best-kept secret" in computing. Well, I think that in the past 10 months, the secret has begun to get out: John Ousterhout has founded Scriptics, devoted to scripting languages; several books have come out; and a Tcl/Tk Consortium has been formed <http://www.tclconsortium.org>. Most significantly, however, there is a *Tcl for Dummies*.

Schroeder and Doyle's *Interactive Web Applications with Tcl/Tk* is a better place for the self-styled "dummy" to start. It is designed for the beginner and does a fine job of explaining applets, widgets, and server-based applications. The accompanying CD-ROM contains Tcl/Tk and the Spynergy Toolkit.

Zeltserman and Puoplo have turned out a useful book on net management tools, *Building Network Management Tools with Tcl/Tk*. As this is the sort of thing that Tcl is ideal for, a reasonable amount of popularization of the language and its toolkit may ensue. I hope so.

Nests, Puffins, and Woodpeckers

Three O'Reilly books with "Net" concerns are on my desk. They all present the competent face all of us have come to expect from ORA. I found Charlie Scott et al.'s *Virtual Private Networks* a fascinating read. Alan Schwartz's *Managing Mailing Lists* covers Majordomo, LISTSERV, ListProc, and SmartList. It should become a "must read" for any sysadmin or site manager. Scott Oaks's *Java Security* is the best of the books on this topic that I've seen.

Dialing for Micropayments

Designing Systems for Internet Commerce is a welcome relief after over a dozen books discussing the question but lacking any real content. Treese and Stewart have written a readable volume that serves as both a technical and a business guide to constructing systems that are maintainable, functioning, and secure. I liked their walk-through of a real system.

It's interesting that the best books in this area have been written by real participants, not by professional writers. (Both Treese and Stewart are with OpenMarket; Dan Lynch is with Cybercash.)

Revivals

Graham's *Sourcebook* is for HTML 4.0 and its extensions. In the past three years this useful book has gained a lot of weight and now tips the scales at 600 pages.

Lippman's *C++ Primer* has gained a coauthor, Josee Lajoie, and many features in this third edition. It is also double the size of the 1991 edition. This is an extraordinary piece of work, reflecting the ANSI/ISO "final draft standard."

Finally, I've received the second edition of Knuth's *Art of Computer Programming*, but had no time to read it. I hope to devote a decent amount of space to the three volumes in a future column.

A Blatant Plug

Over the past two years I have been working on a four-volume *Handbook of Programming Languages*. By the time you read this, it should be available. Get your academic or corporate libraries to buy them.

More books reviewed:

Dave Zeltserman & Gerard Puoplo

Building Network Management Tools with Tcl/Tk

Upper Saddle River, NJ: Prentice Hall, 1998.
ISBN 0-13-080727-3. Pp. 429.

Charlie Scott, et al.

Virtual Private Networks

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-319-7. Pp. 177.

Alan Schwartz

Managing Mailing Lists

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-259-X. Pp. 282.

Scott Oaks

Java Security

Sebastopol, CA: O'Reilly & Associates, 1998.
ISBN 1-56592-403-7. Pp. 456.

G. Winfield Treese & Lawrence C. Stewart

Designing Systems for Internet Commerce

Reading, MA: Addison Wesley, 1998.
ISBN 0-201-57167-6. Pp. 375.

Ian S. Graham

HTML 4.0 Sourcebook rev. edition

New York: John Wiley, 1998.
ISBN 0-471-25724-9. Pp. 631.

Stanley B. Lippman & Josee Lajoie

C++ Primer, 4th ed.

Reading, MA: Addison Wesley, 1998.
ISBN 0-201-82470-1. Pp. 1237.

book reviews

Mark Harrison

Tcl/Tk Tools

O'Reilly & Associates 1997. ISBN 1-56592-218-2. Pp. 678. $49.95.

Reviewed by Clifton Flynt
<clif@cflynt.com>

One of the best things about the Tcl/Tk language is the large number of packages that have been developed around it. Some of these extensions have used Tcl as a base language for creating special-purpose tools; others have extended the Tcl language with new general-purpose commands.

Learning how to use the packages hasn't always been easy, though. Most of them include a set of man pages, but unfortunately, man pages aren't always sufficient. Tutorial material and examples have been sparse.

Tcl/Tk Tools steps into that gap and fills it well. Most of this book consists of tutorial and overview chapters on several of the more popular Tcl/Tk extensions. There is also a chapter of techniques for debugging Tcl scripts, instructions for merging multiple packages into a customized shell, and instructions on configuring X window system security using xauth.

The CD-ROM included with the book has binaries for Solaris and Linux, and source code for the packages discussed in the book. The code distributed on the CD-ROM is Tcl 7.6 based, not Tcl 8.0, but having the proper revisions of the sources in one place makes compiling the packages a painless task. The other disappointment is that the CD-ROM doesn't include the examples.

This is not a book for the Tcl/Tk novice. It is not a Tcl/Tk tutorial. It assumes that you already speak Tcl/Tk and need to learn how to work with an extension package.

The book covers several types of extensions: general programming, GUI, and specific application extensions. This breadth of coverage makes it very likely that any Tcl programmer will find something of use in it.

The general-purpose Tcl programming extensions include Tcl-DP, for distributed processing; TclX, with many new Tcl commands; and Expect, the extension that makes it easy to control other programs from Tcl.

The graphics extensions covered include BLT, which has a graphing widget that cured me of gnuplot; TIX, with more GUI widgets than I could ever use; and [incr Tk] and [incr Widgets] for those who prefer object-oriented programming techniques.

The special-purpose applications covered include TSIPP, a front end to the SIPP graphics-rendering library that is lots of fun (but you can burn up a lot of cycles playing with it); GroupKit, for building groupware packages; and SybTcl and OraTcl, the Tcl front ends to the Sybase and Oracle database libraries.

Each of the chapters is actually written by a different author, usually the author of the package. Most of the chapters stand alone; thus, you can pick up the book and read chapters at random. The exceptions to this are the second and third chapters on [incr Tk] and [incr Widgets], respectively, which assume you're already familiar with [incr Tcl], which is covered in the first chapter.

As with any work of multiple authors, the style and content varies from chapter to chapter. Some are sufficiently detailed tutorials that you can start using the package with no other reading. Others provide more of a high-level introduction that gets you started and makes the man pages accessible.

I found that I could use the BLT package after reading George A. Howlett's chapter without needing to check the man pages for more details.

Don Libes points out in his chapter on Expect that an entire book could be written about Expect (which is what he did), but he provides a good introduction to the more commonly used commands. I could start writing useful programs after reading his chapter, but would need more information to write a major project with Expect.

Mark Diekhans's chapter on TSIPP didn't give me the information I needed to create the nifty images I wanted to create, but it gave me the overview I needed to understand the man pages. Man pages provide a lot of detailed information, but they usually don't tell you where to start or provide the examples that show how to use a package. I wouldn't have been able to start using TSIPP without this chapter.

The other chapters in the book left me feeling secure that I could sit down and start writing code, although I haven't had the time to prove the fact empirically. Overall, this is an excellent book. I highly recommend it to anyone working with Tcl/Tk.

Craig Hunt

TCP/IP Network Administration, 2nd Ed.

O'Reilly & Associates, 1998. ISBN 1-56592-322-7. Pp. 612 (184 pages of index and appendices). $32.95 Paper.

Reviewed by Rob Jenson
<robjen@spotch.com>

Craig Hunt describes the target audience of his book in the preface to the first edition of *TCP/IP Network Administration*. This book is for neither UNIX dummies nor idiots. Nor is it for network administration gurus and geniuses. Most other system administrators who deal with a system connected to a TCP/IP network can get something out of *TCP/IP Network Administration*.

The book gives an overview of and some details about the various components that network an individual UNIX system to a TCP/IP network. Some of the

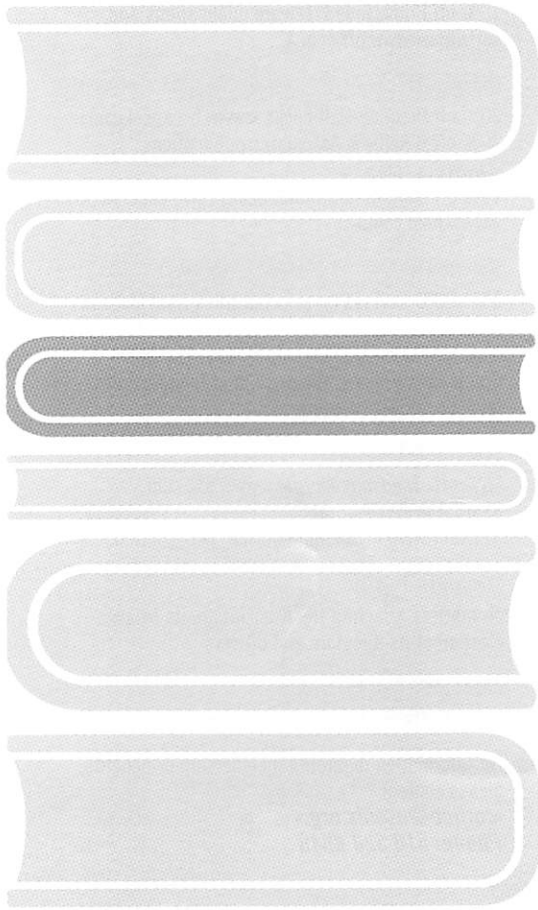
abstract basics of networking and TCP/IP are covered, followed by an introduction to the important network services. The author then takes you, step by step, through a detailed discussion of the configuration of various network components. Every section provides a pointer to the appropriate O'Reilly book that is specific to the topic at hand, if one exists. Absent are pointers to some of the canonical sources of the source code and official documentation sites for some of the services described. My expectation from a paperback book that is such a reference would be that pointers to some of the basic sites would be included, especially since many TCP/IP service programs are in a constant state of flux.

Craig Hunt gives you many command-line examples to describe what you want to do. If you are working with Solaris or Linux operating systems, this is a miniature cookbook of how to perform many of the network administration tasks for those systems. There are also some notes on configuring a BSD kernel. Linux and Solaris are two of the more common flavors of UNIX in use today, but neither is a contender for "vanilla." For other flavors of UNIX, you will be able to find the appropriate manual pages online after reading the book.

The appendices are voluminous, and, at first glance, excessive. Specific configuration guides for PPP, gated, named, dhcpd, and sendmail are available from the suppliers of those respective packages on the Internet. Eventually those will be your bible for configuring and troubleshooting the software. However, Craig Hunt's appendices cover most of the bases in a very consistent, linear fashion that is written from the perspective of someone who has used the software frequently. He will get you started, he will probably get you up and running, and he'll definitely leave you ahead of the game when you need to dig deeper.

TCP/IP Network Administration is a well-written book. It reads smoothly from beginning to end. It was a useful overview of many host networking elements of systems administration that I've learned over the past ten years. The network security and the troubleshooting sections were a disappointment to me. I would have liked to see more information about how to troubleshoot specific problems with each section. That also holds true for the security concerns attached to many of the network services. You will not find anything here about BIND 8 or IPv6 and very little about hardware. If your idea of network administration has anything to do with configuring router hardware, load-balancing traffic, designing LAN, CAN, or WAN infrastructures, or choosing high-speed switch technology, this book has nothing to interest you. Everything here is about how to make UNIX play nicely with other computers on the network, once you have your ducks in a row with the network itself.

So, who will find this book useful? I would certainly recommend it as required reading for any novice or junior level systems administrator. It probably has a place on the bookshelf of many intermediate/advanced or senior level system administrators. It's also a good loaner for the curious power users who want to learn more about what is involved in keeping their network fed and happy. As a reference guide it has a limited shelf life, because the details change so quickly. If I were going to be suddenly shipped off-planet as a new system administrator and I could take nothing with me but three books, *TCP/IP Network Administration* would be worth considering as one of them.



USENIX news

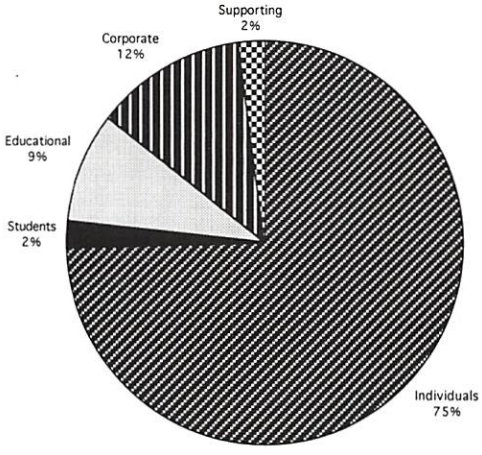
Member Dues

by Ellie Young
Executive Director
<ellie@usenix.org>

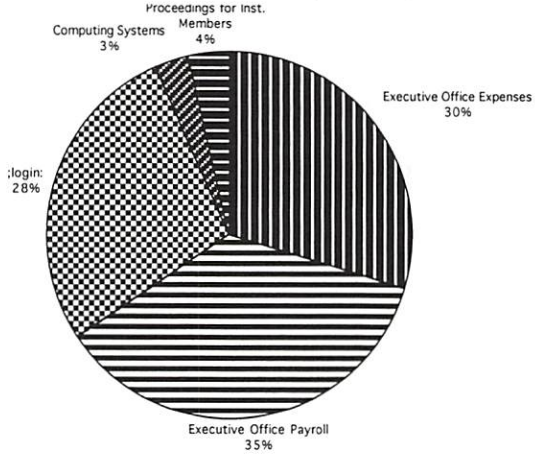
Here are a few charts that might help those of you who are curious about how your USENIX and SAGE dues are spent. The first shows the total dues income ($580,000 in 1997) divided by type of membership. The second chart then presents how those dues are spent. Note that income from our conferences cover all costs of the conference office, exhibition and marketing. The third chart shows how the executive office spends its money. The "other" category covers such items as taxes, licenses, bank charges and miscellaneous expenses. The fourth chart indicates where most of the money allocated to standards activities and good works are spent ($750,000 in 1997). (See "There's Gold in Good Works: A Report on USENIX Support of Worthwhile Projects" on page 72 of the February 1998 issue of *;login:* and on the USENIX Web site at <http://www.usenix.org/about/goodworks.html>. These funds come from the income generated by the USENIX conferences and interest income from the Association's reserve fund.

Two charts deal with SAGE income ($148,000 in 1997) and direct expenses. Allocated expenses (staff and overhead) are not reflected in the direct expenses chart.

Membership Income Sources



Where Did Your '97 Membership Dues Go?



USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,
the Association's magazine, published six to eight times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java, and C++, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to papers
from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web <http://www.usenix.org>.

Discounts on registration fees
for all USENIX Conferences, as many as eight every year.

Discounts
on the purchase of proceedings and CD-ROMS from USENIX conferences

PGP Key Signing service
available at conferences.

Discounts
10% off BSDI, Inc. products.
<www.bsdi.com>.

Discounts
10% off Prime Time Freeware publications and software <www.pff.com>

Discounts
20% off Prentice-Hall books <www.bookpool.com>
20% off O'Reilly & Associates publications <www.ora.com>
10% off Morgan Kaufmann books (give code :ALOG) <www.mkp.com>

Savings
10%-20% savings on selected titles from McGraw-Hill <www.books.mcgraw-hill.com>, John Wiley & Sons <www.wiley.com/compbooks>, The Open Group <www.opengroup.org>, and the MIT Press (give code UNIX1) <mitpress.mit.edu>

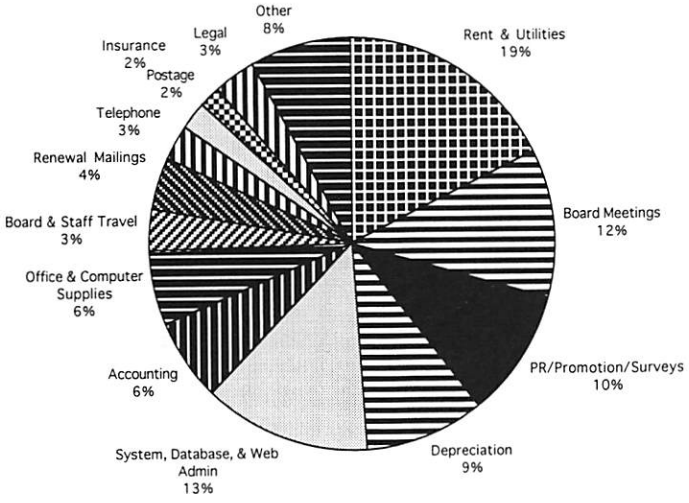
Special subscription rates
15% off *The Linux Journal* <www.ssc.com>
$5 off *The Perl Journal*
<orwant.www.media.mit.edu/the_perl_journal>

The right to vote
on matters affecting the Association, its bylaws, election of its directors and officers

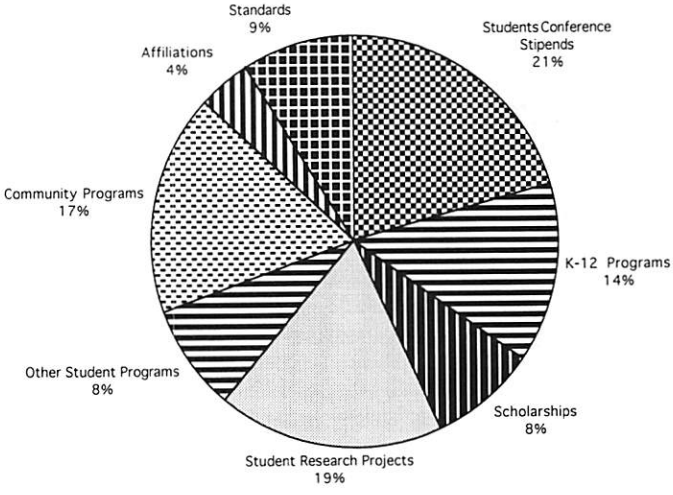
Optional membership
in SAGE, the System Administrators Guild

For information regarding membership or benefits, please contact
<office@usenix.org>
Phone: 510 528 8649

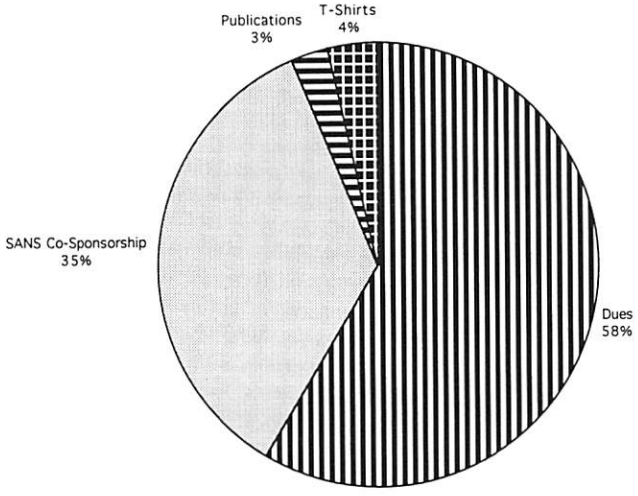
Executive Office Expense



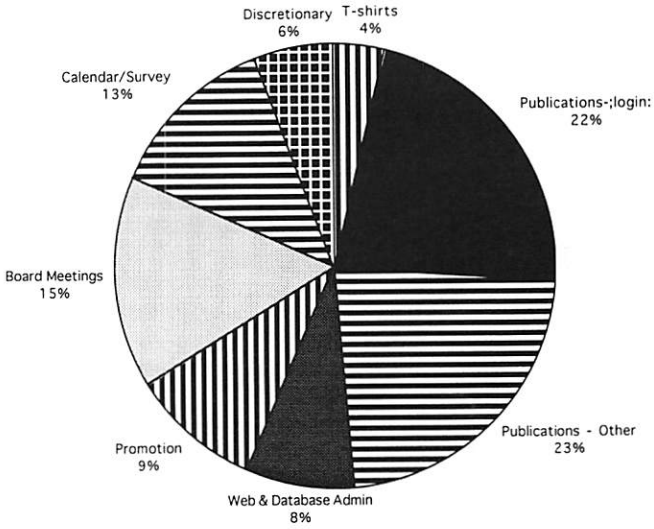
Good Works and Standards Activities



1997 SAGE Income Sources



1997 SAGE Direct Expenses



1997 Financial Statements

STATEMENT IN REVENUE AND EXPENSES AND CHANGES IN NET ASSETS

For the Thirteen Months Ended December 31, 1997
and the Year Ended November 30, 1996

	1997	1996
REVENUE		
Membership dues	$ 580,238	$ 509,249
Product sales	63,771	69,599
Conferences/Workshops	3,649,571	3,519,268
SAGE	201,016	144,629
Interest on operating funds	76,426	45,166
Other non operating income & expense	-29,590	15,942
Total revenue	$ 4,541,432	$ 4,303,853
EXPENSES		
Executive office/General admin	$ 574,905	$ 444,934
Membership, ;login:, web site	242,633	213,030
Conferences/Workshops	2,663,607	2,452,129
SAGE expenses	148,117	72,705
Product expenses	36,582	45,585
Projects & Good Works	750,112	244,543
Depreciation	38,967	24,352
Total operating expenses	$ 4,454,923	$ 3,497,278
Net operating surplus	86,509	806,575
Net investment income	830,096	1,574,893
Excess of revenue over expense	916,605	2,381,468
Restricted Fund/STUG Award	621	
Net Assets, beginning of year	5,296,822	2,915,354
Net Assets, before unrealized gain	$ 6,214,048	$ 5,296,822
Unrealized gain on securities	557,218	1,157,189
Net Assets, end of year	$ 6,771,266	$ 6,454,011

BALANCE SHEET

As of December 31, 1997 and November 30, 1996

ASSETS	1997	1996
Current Assets		
Cash	$ 1,749,006	$ 1,791,534
Receivables	48,897	50,194
Prepaid Expenses	131,996	163,278
Inventory	30,324	37,446
Total Current Assets	1,960,223	2,042,452
Investment in Securities	5,008,774	4,862,184
Net Property and Equipment	126,393	96,566
Total Assets	$ 7,095,390	$ 7,001,202
LIABILITIES & NET ASSETS		
Liabilities		
Accrued Expenses	$ 123,511	$ 184,691
Deferred Revenue	200,613	362,500
Total Liabilities	$ 324,124	$ 547,191
NET ASSETS	6,771,266	6,454,011
TOTAL LIABILITIES & NET ASSETS	$ 7,095,390	$ 7,001,202

STATEMENT OF CASH FLOWS

For the Thirteen Months Ended December 31, 1997
and the Year Ended November 30, 1996

	1997	1996
Cash flows from operating activities		
Excess of revenue over expense	$ 916,605	$ 2,381,468
Depreciation	38,967	24,352
Decrease/(Increase) in receivables	1,297	-19,986
Decrease/(Increase) in inventory	7,122	-473
Decrease/(Increase) in prepaid expense	31,282	-14,670
Increase/(Decrease) in accrued expenses	-61,180	100,244
Increase/(Decrease) in deferred revenue	-161,887	226,285
Total	$ -144,399	$ 315,752
Net cash provided by operating activiti	$ 772,206	$ 2,697,220
Cash flows provided by/(used for) investing activities:		
Purchase of investments	$ -746,561	$ -3,548,828
Addition to STUG fund	621	
Sale of investments		1,784,338
Purchase of property & equipment	-68,794	-43,669
Net cash used for investing activities	$ -814,734	$ -1,808,159
Net change in cash & equivalents	-42,528	889,061
Cash & equivalents, beginning of year	1,791,534	902,473
Cash & equivalents, end of year	$ 1,749,006	$ 1,791,534

USENIX PGP Key Signing Service to be Discontinued

by Greg Rose

Greg Rose is Vice President of USENIX.

<ggr@usenix.org>

For over two years, USENIX has been running a PGP key signing service, in which people could present identification at a USENIX conference and subsequently have their PGP key signed by a well-known USENIX key, thus becoming connected to the PGP "Web of Trust." This service had the innovative feature that individuals did not have to have their PGP keys ready in advance. A number of reasons have contributed to the decision to discontinue the service:

- The service began to take on the trappings of a certification authority. While there is nothing inherently wrong with that, there are others out there who perform exactly that service. USENIX's goal was to enhance connectivity within the Web of Trust, not to move it further toward a hierarchical organization.
- New, and mutually incompatible, versions of PGP significantly complicated the programming, and hence the cost, of the service.
- The protocol that enabled trustworthy operation of the service turns out to have many unexpected failure modes. That is, when it works, it works fine, but when any mistake is made, it fails in ways that break the automated scripts . . . and newer versions of PGP with increased emphasis on interactive operation try to be "helpful" with disastrous results. Because the service was oriented to helping newcomers, only about one-third of the submissions went through without error the first time; others needed to be retried, required manual intervention, or simply had to be rejected. (For example, about 20% of submissions used the example secret from the documentation, and not the issued secret!)

The signatures already made are still valid, the Web interface for checking them will continue to work indefinitely, and it will continue to be possible to communicate with USENIX using PGP. (Note that for correspondence the office now supports both RSA and DH/DSS keys).

To replace the service, USENIX intends to provide some support for PGP Keysigning BoFs, and to do it in such a way that outside parties can also make use of many of the features. This furthers the goal of enhancing the Web of Trust without the overhead of a certification authority.

The useful feature of enabling people to come unprepared is going to be continued by a project we are calling "torn money" (after the old spy films where someone's identity was established by them having half of a torn banknote). A preliminary version of this was available at the New Orleans general conference, with further Web-based support coming soon. In this, individuals at the PGP Keysigning BoF will be helped with sheets of "shared secrets" which can be given to others to forge a link after they have left the BoF. Watch the USENIX PGP Web page for more details as they become available.

Riding in Style

Ted Dolotta, who since 1981 was the proud owner of the California Department of Motor Vehicles "UNIX" license plates, has just moved back to New Jersey. Since he could no longer use the plates, he decided to auction them off, with USENIX student programs to receive the proceeds.

The auction, which Ted announced on the USENIX Web site and other venues and conducted by email (regularly advising all the bidders of the current high bid), ended at midnight on July 14. Over 30 participants competed for the prize.

The top bid – $6,000 – came from John Mashey, author of the Mashey Shell (the predecessor of the Bourne and Korn Shells) and co-author of the MM nroff/troff macro package. John is now a Chief Scientist at Silicon Graphics.

Says John, "As I am recovering (very well) from a heart bypass operation in late June, some may attribute this action on my part to drug-induced craziness, but really, I waited until I was off the painkillers before bidding, and besides, it's a great excuse to buy a new car."

USENIX thanks John, Ted, and all the other bidders for this generous gift.

1998 US Open USACO Programming Championship

by Rob Kolstad

Rob Kolstad, editor of *;login:*, is head coach of the USA Computing Olympiad Team.

<kolstad@usenix.org>

One hundred twenty-nine entrants (each of whom had submitted all solutions) were graded for the US Open held April 8–13, 1998. US entrants participated in the contest in a proctored, five-hour, examination-style environment. Six domestic finishers scored 600 points or more:

Adrian Sox	916
Matt Craighead	767
Benjamin Mathews	667
Brendan Connell	654
John Danaher	641
Reid Barton	618

Adrian Sox was the contest's overall winner, with last year's champion, Matt Craighead, in the runner-up position.

Fourteen international participants (in non proctored exams) scored above 600 points. Andy Kurnia, Indonesian software superstar, was challenged by veteran Timo Burkard and newcomer Peter Hutagalung, but held on for first place in the international division:

Andy Kurnia, Indonesia	865
Timo Burkard, Germany	844
Peter Hutagalung, Indonesia	831

International contestants hailed from seventeen countries, with almost three-quarters of the total entries coming from the USA – way up from previous contests.

The problems were much more challenging for this contest, as befits the final contest of the year. Here's one of the two most difficult problems. Greg Galperin from MIT created the marvelous phrasings of a recast travelling salesman problem. The interesting part is the time limit – contestants had to create a number of heuristics (within the five-hour coding time limit) to get "good" solutions.

Avoiding les Entarteurs

Billy Goats, chairman and CEO of industry giant Mycowsoft, is visiting his European branch offices. But alas, the notorious Le Gloupier is at it still, and has placed his agents (the entarteurs) in the streets of Europe, waiting to throw cream pies in celebrities' faces. Luckily, Mycowsoft Corporate Intelligence has determined how many entarteurs lie in wait along each of the routes between offices and has asked you to plan Billy's visits so as to minimize the number of pies Billy has to wipe off his face.

The first line of the input file will contain N, the number of corporate offices that must be visited, 1 <= N <= 50. Each of the next N lines is a space-separated list of N integers between 0 and 1,000 inclusive, where on the i-th line in the block the j-th number indicates the number of cream pie-throwing pranksters who are waiting along the route from office i to office j. (Restated, this number is on the i-th row and j-th column of the NxN grid of numbers, where the i-th row in the block is on the i+1-st line of the file.)

Billy needs to visit all N of Mycowsoft's corporate offices, each one exactly once, starting from any of them and ending in any other. You should create such a route and print the office numbers visited in order on the first line of the output file separated by spaces (there should be N of them), and on the second line of the file print the total number of entarteurs encountered on that route. Your solution will be scored based on how many entarteurs were encountered; the better (smaller) that number is, the more points you get for that test case.

The problem must run within a five second time limit.

News from EurOpen.SE

By Jan Saell

Jan Saell is chairman of EurOpen.SE.

<jan@yask.com>

EurOpen.SE (the Swedish Association of UNIX users) is now a fully affiliated member of USENIX. This seems to have had a really positive influence on our organization. After some years with declining membership and not much going on in the organization, a small but a significant number of people are becoming new members.

By targeting system administrators, and especially by not being afraid to talk about the "Evil Empire" (Microsoft) both USENIX and Europen.SE are moving in helpful directions. Europen.SE has long supported UNIX users who are now mostly senior administrators; they really need information about both UNIX-related issues and all the new stuff that comes with NT and Windows 95/98.

The Annual Meeting

EurOpen.SE will hold its annual meeting October 7-8, 1998. The meeting will take place on the boat from Stockholm to Åbo in Finland. Speakers will be covering:

- Backup/restore of large systems
- Cluster solutions to create high availability servers
- High availability hardware and software.
- Network redundancy and fault tolerance
- New trends and information in the UNIX/Open System area

Vendor presentations from Digital, Sun, SCO, Legato, and others will be among the talks. See <http://www.europen.se> for more information about the meeting.

The 1st Nordic USENIX/EurOpen Conference (NordU99)

NordU99 will take place February 9-12, 1999, at the Grand Hotel in Stockholm. Two days of tutorials and two days of conference sessions will cover:

- Security tools and techniques
- Electronic commerce
- Electronic publishing
- Innovative system adminstration tools and techniques
- Performance analysis, monitoring, and tuning
- Networking

If you have interesting work to present in these areas, please send an extended abstract as soon as possible.
The call for papers is online at <http://www.europen.se/NordU99>. August 14 is the deadline for submissions.

Our goal is for this to be an annual conference for all UNIX/Open System people in the Nordic area – not just in Sweden.

We'll keep you up to date on Europen.SE activities and hope to see some of you at NorU99.

WEB SITE

http://www.usenix.org

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS

Eileen Cohen
Telephone: 510 528 8649
Email: <cohen@usenix.org>

USENIX SUPPORTING MEMBERS

ANDATACO
Apunix Computer Services
Auspex Systems, Inc.
Cirrus Technologies
CyberSource Coporation
Digital Equipment Corporation
Earthlink Network, Inc.
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
Invincible Technologies Corporation
Lucent Technologies, Bell Labs
Motorola Global Software
Nimrod AS
O'Reilly & Associates
Performance Computing Magazine
Sun Microsystems, Inc.
UUNET Technologies, Inc.
WITSEC, Inc.

Twenty Years Ago in ;login:

by Peter H. Salus

Peter H. Salus is the author of *A Quarter Century of UNIX* (1994) and *Casting the Net* (1995). He has known Lou Katz for over 40 years.

<peter@pedant.com>

The AT&T attorneys don't seem to have had much impact on the August 1978 *;login:*, which was subheaded "THE UNIX NEWSLETTER." The entire front page of the issue was devoted to a brief, neat announcement:

> The special issue of *The Bell System Technical Journal* devoted to the UNIX system will be available on or about August 1, 1978. It is Part 2 of Vol. 57, No. 6 (July-August 1978).
>
> Bell Labs plans to send a complimentary copy to each licensee of UNIX, MINI-UNIX, and PWB/UNIX.
>
> Additional copies cost $1.50 each, plus 15 cents per copy for foreign postage . . . .

This was the light blue UNIX issue. It began with a foreword by McIlroy, Pinson, and Tague, proceeded to Ritchie and Thompson's 1974 *CACM* article, to Thompson's "UNIX Implementation," and to pieces by (among others) Bourne, Johnson, Lesk, Kernighan, Lycklama, Ossanna, Cherry, Morris, Dolotta, Haight, and Mashey. Four hundred extraordinary pages for $1.50, plus $0.15 for foreign postage. Wow! Can you even get a magazine for $1.50 20 years later?

The September 1978 issue was the last one for a long time. I will reserve that tale for a later chronicle and look at the contents now.

A lot of folks wrote letters to Mel Ferentz in those days: remember, USENIX was the sole source for bug fixes, application software, etc. And although the source code was AT&T's "property," the fixes, tweaks, and improvements were, for the most part, "open." In fact, while the OS was what Eric Raymond would call a cathedral, the remainder was a bazaar.

Tom Ferrin wrote, describing the Computer Graphics Lab at UCSF and offering both the technical report (they supported about 12 users in realtime: "<= 50 msec") and their graphics subroutine package "(both are free of charge)."

Eric A. Brooks, from Analytics in Saskatoon, wrote that they had an RK06 driver and were considering an RM03 driver, but they didn't want to "reinvent the wheel." So if there was one, could he be told of it. (There was a note from the University of Guelph asking about DU 11 and DQ 11 drivers.) And there was a letter from Thomas A. Berson (Ford Aerospace) offering "a Formal Specification for 6.0 UNIX."

This was interesting. But appended was a 14-page document: "PROPOSED BYLAWS OF USENIX." It began: "The name of the organization is Usenix." It had been 14 months since *UNIX NEWS* had been pushed by AT&T to become *;login:*. It was a month more since the appointment of the "USENIX committee." Mel Ferentz, Mars Gralia, Lou Katz, Lew Law, and Peter Weiner had delivered on their assignment. But there was much more to come.

3rd USENIX Workshop on Electronic Commerce

August 31 –September 3, 1998, Tremont Hotel, Boston, Massachusetts

Tutorial Program, Monday, August 31

Smart Cards,
Scott Guthery, CertCo

Using Cryptography,
Bruce Schneier, Counterpane Systems

Setting up and Maintaining a Secure Web Server,
Bryan Buus, XOR Network Engineering

The Law of Electronic Commerce—
Contracts, Records, and Privacy,
Benjamin Wright, Attorney and Author

Cryptography for the Internet,
Bruce Schneier, Counterpane Systems

Electronic Payment Systems, *Clifford Neuman, USC*

Technical Program, September 1–3

***Including sessions on Public Key Infrastructure (PKI)**

Tuesday, September 1

Keynote Address

Research Directions in Electronic Commerce
Stuart Feldman, *IBM Institute for Advanced Commerce, IBM T.J. Watson*

Advances in Payment Technology

Electronic Commerce and the Street Performer
Bruce Schneier, John Kelsey, *Counterpane Systems*

VarietyCash: A Multi-Purpose Electronic Payment System
Mihir Bellare, *University of California, San Diego;* J. Garay, C. Jutla, *IBM T.J.Watson Research Center;* M.Yung, *CertCo*

NetCents: A Lightweight Protocol for Secure Micropayments
Tomi Poutanen, Michael Stumm, *Univ. of Toronto,* Heather Hinton, *Ryerson Univ.*

*Public Key Implementation Case Study

Presenter: Juan Rodriguez-Torrent, *IBM Corporation,* and others from *National Automated Clearing House Association (NACHA)*

Respondent: Steve Cohen, *ncipher Inc.*

Auction Markets

The Auction Manager: Market Middleware for Large-Scale Electronic Commerce
Tracy Mullen, Michael P. Wellman, *Artificial Intelligence Laboratory*

Internet Auctions
Manoj Kumar, Stuart I. Feldman, *IBM T. J. Watson Research Center*

Electronic Auctions with Private Bids
J. D. Tygar, Michael Harkavy, *Carnegie Mellon University;* Hiroaki Kikuchi, *Tokai University*

Patent Panel: Intellectual Property and Electronic Commerce

Wednesday, September 2

*PKI Session: Trust Models

Presenter: Paul Van Oorschot, *Entrust Technologies*

Respondent: Bill Frantz, *Electronic Communities*

Secure Systems – What It Takes

A Resilient Access Control Scheme for Secure Electronic Transactions
Jong-Hyeon Lee, *University of Cambridge*

Trusting Trusted Hardware: Towards a Formal Model for Programmable Secure Coprocessors
Sean W. Smith, Vernon Austel, *IBM T.J Watson Research Center*

On Secure and Pseudonymous Client-Relationships with Multiple Servers
Daniel Bleichenbacher, Eran Gabber, Phil Gibbons, Yossi Matias, Alain Mayer, *Lucent Technologies, Bell Laboratories*

Digital Bearer Settlement and the Geodesic Economy

Robert Hettinga, *Philodox Financial Technology Evangelism*

*Electronic Commerce Needs No PKI

Presenter: Win Treese, *Open Market, Inc.*

Respondent: Joan Feigenbaum, *AT&T Labs—Research*

Deployable Internet/Web Services

Secure WWW Transactions Using Standard HTTP and Java Applets
Bruno Crispo, *University of Cambridge*; F. Bergadano, M. Eccettuato, *Universita di Torino Italy*

A Generalized Digital Wallet Architecture
Neil Daswani, Dan Boneh, Hector Garcia-Molina, Steven Ketchpel, Andreas Paepcke, *Stanford University*

The Eternal Resource Locator: An Alternative Means of Establishing Trust on the World Wide Web
Ross J. Anderson, Václav Matyás Jr., Fabien A. Petitcolas, *Cambridge University*

Detecting Hit Shaving in Click-Through Payment Schemes
Michael Reiter, *AT&T Labs—Research*

Thursday, September 3

Consumer Service

Sales Promotions on the Internet
Manoj Kumar, Quoc-Bao Nguyen, Colin Parris, Anant Jhingran, *IBM T. J. Watson Research Center*

General-Purpose Digital Ticket Framework
Ko Fujimura, Yoshiaki Nakajima, *NTT Information and Communication Systems Labs*

Towards a Framework for Handling Disputes in Payment Systems
N. Asokan, Els Van Herreweghen, Michael Steiner, *IBM Research Laboratory*

*Current Mapping of PKI to the Law

Presenter: Dan Greenwood, *Commonwealth of Mass.*

Respondent: Jane Winn, *Southern Methodist University School of Law*

*Name-Centric vs. Key-Centric PKI

Key-centric Presenters: Carl Ellison, *CyberCash Inc.*, and Perry Metzger, *Piermont Information Systems Inc.*

Name-centric Presenters: Warwick Ford, *Verisign Inc.*, and Steve Kent, *CyberTrust Solutions, GTE*

Short Talks/Works-in-Progress Reports (WIPs)

6th Annual Tcl/Tk Conference

September 14-18, 1998, Paradise Point Resort, San Diego, California

Tutorial Program, September 14–15

Database Programming with Tcl/Tk

Effective Tcl/Tk Programming

Cross-Platform Development

CGI Scripting

Building Client/Server Applications

The Dark Secrets of Tcl Development: Debugging, Testing and Packaging

Tcl Extension Building and SWIG

Object-Oriented Programming with [incr Tcl]

New Features in Tcl 8.0 and Tcl 8.1

Everything Your Mother Never Told You About ClientData: C Programming With Tcl/Tk

Building Mega-Widgets with [incr Tk]

Tcl and Java Programming: Practice and Pitfalls

Technical Program, September 16–18

Wednesday, September 16

Keynote Address : Tcl/Tk, Agents, and Makin' Pictures: A Whirlwind Tour

Dr. Michael B. Johnson, *Pixar Animation Studios*

Applications

NBC's GEnesis Broadcast Automation System: From Prototype to Production

S. J. Angelovich, *NBC Broadcast and Network Operations*, K. B. Kenny, B. D. Sarachan, *GE Corporate Research & Development*

An Extensible Remote Graphical Interface for an ATM Network Simulator

Michael D. Santos, P. M. Melliar-Smith, L. E. Moser, *University of California, Santa Barbara*

WinACIF: A Telecom IC Support Tool Using Tcl/Tk

David Karoly, Todd Copeland, David Gardner, *Advanced Micro Devices*

Charity Telethon Supported by Tcl/Tk

Dave Griffin, *Digital Equipment Corporation*

Tcl/Tk Update

John Ousterhout, *Scriptics Corporation*

Object Technology

The Tycho Slate: Complex Drawing and Editing in Tcl/Tk

H. John Reekie, Edward A. Lee, *University of California, Berkeley*

Iclient/Iserver: Distributed Objects using [incr Tcl]

Lee F. Bernhard, *Bell Labs Innovations for Lucent Technologies*

Data Objects

George A. Howlett, *Bell Labs Innovations for Lucent Technologies*

Thursday, September 17

Invited Talk: Tcl/Tk for Dummies

Tim Webster, *Timothy Webster Design and Consulting, Inc.*

Testing and Debugging

A Tcl-Based Multithreaded Test Harness

Paul Amaranth, *Aurora Group, Inc.*

Using Tcl/Tk for an Automatic Test Engine

Allen Flick, *DSC Communications Corporation*, James Dixson, *Silicon Valley Networks Corporation*

wshdbg - A Debugger for CGI Applications

Andrej Vckovski, *Netcetera AG*

Web Technology (Server-Side)

Session Chair: John Reekie, *University of California, Berkeley*

NeoWebScript: Enabling Webpages With Active Content Using Tcl

Karl Lehenbauer, *NeoSoft, Inc.*

TclXML: XML Support For Tcl

Steve Ball, *Zveno Pty Ltd*

Creating High Performance Web Applications using TCL, Display Templates, XML, and Database Content

Alex Shah, Tony Darugar, *Binary Evolution, Inc.*

Conference Lunch: Tcl/Tk and the Languages of the Net: 1971-1998

Speaker: Peter Salus, *The Tcl/Tk Consortium*

Web Technology (Client-Side)

WebWise Tcl/Tk: A Safe-Tcl/Tk-based Toolkit Enhanced for the World Wide Web

Hemang Lavana, Franc Brglez, *North Carolina State University*

Internet-Based Desktops: Collaborative and Recordable

Amit Khetawat, Hemang Lavana, Franc Brglez, *North Carolina State University*

Creating A Multimedia Extension for Tcl Using the Java Media Framework

Moses Dejong, Brian Bailey, Joseph Konstan, *University of Minnesota*

Visualizing Personal Web Caches with Caubview

Charles L. Brooks, Murray S. Mazer, and Frederick Hirsch, *The Open Group Research Institute*

Work-in-Progress Sessions (WIPs)

Poster Session/ Informal Demonstrations

Friday, September 18

Invited Talk: Yacc Meets Tk?

Steve Johnson, *Transmeta Corporation*

Language Issues

Using Content-Derived Names for Package Management in Tcl

Kennedy Akala, Ethan Miller, *University of Maryland, Baltimore County*, Jeff Hollingsworth, *University of Maryland*

Using Tcl to Rapidly Develop a Scalable Engine for Processing Dynamic Application Logic

Greg Barish, *Healtheon Corporation*

Using Tcl to Script CORBA Interactions in a Distributed System

Michael Miller, *Advanced Micro Devices*, Srikumar Kareti, *Honeywell Technology Center*

Panel

Tcl in the Bazaar – A United Front

Moderator: Michael McLennan, *Bell Labs Innovations for Lucent Technologies*

Panelists: Mark Harrison, *AsiaInfo Computer Networks*, Beijing; Peter Salus, *The Tcl/Tk Consortium*; Brent Welch, *Scriptics Corp.*

CONFERENCE URL: *http://www.usenix.org/events/tcl98*

1st International SANE Conference

November 18–20, 1998, Maastricht, The Netherlands
http://www.nluug.nl/events/sane98/

The program for SANE '98, an international conference on System Administration and Networking, focused on UNIX, is now available. Please go to http://www.nluug.nl/events/sane98/ to find renowned speakers for many interesting topics. SANE '98 is the place where you will hear, discuss, and then put to use the latest research, approaches, tools, and techniques for practical system administration and security.

We find ourselves in exciting times where the daily email loads rapidly increase (can your ISP's mail transfer agent handle zillions of messages day by day?), where spamming burns valuable resources, where security attacks become more sophisticated, and where the Open Source movement shows remarkable progress.

The conference kicks off on Wednesday, November 18, with the opportunity for in-depth study! Choose among three tracks of tutorials (covering performance tuning, security, IPv6, and general UNIX systems administration) led by experienced and respected instructors: Bill Cheswick, Adrian Cockcroft, John van Krieken, Walter Belgers, Hans van de Looy and Evi Nemeth.

During the second and third days of SANE '98 you will (after the keynote address) be able to choose from two tracks of interesting presentations of both refereed papers and invited talks. Hear about experience reports, (b)leading edge developments, the use of open source software, and commercial uses of UNIX. You will find a remarkable line-up of speakers, including Wietse Venema, Rob Kolstad, Eric Troan, Phil Zimmermann, Bruce Perens, Brad Knowles and many more. In the late afternoon of the second day of the conference there will be Birds-of-a-Feather sessions.

Thursday and Friday you can also stroll along the exhibition area, where vendors will demonstrate their latest hardware and software products that they hope will help you do your job more efficiently and effectively.

SANE '98 will be held in the Maastricht Exposition and Conference Center, MECC, close to the medieval center of the city of Maastricht, in the south of the Netherlands, close to the borders with Belgium and Germany. Maastricht has excellent train connections with Amsterdam-Schiphol airport (and even the International airport Maastricht-Aachen is close), so reaching SANE '98 from all over the world is pretty easy.

Please join us.

— Edwin H. Kremer and Jan Christiaan van Winkel, *Program Co-Chairs*

http://www.nluug.nl/events/sane98/

Call for Papers USENIX

1st Conference on Network Administration

Sponsored by USENIX, the Advanced Computing Systems Association and Co–sponsored by SAGE, the System Administrators Guild

April 7–9, 1999
Santa Clara Marriott Hotel
Santa Clara, California

Important Dates

Submission deadline: *November 6, 1998*
Notification to authors: *December 1, 1998*
Camera–ready papers due: *February 23, 1999*
Registration material available: *February, 1999*

Program Committee

Chair: David Williamson, *Global Networking and Computing*
Brent Chapman, *Covad Communications*
Paul Ferguson, *Cisco*
Jeff Jensen, *WebTV Networks*
William LeFebvre, *GroupSys*
Bryan McDonald, *GNAC*
Hal Pomeranz, *Deer Run Associates*

Overview

The networking administration community is expanding at an ever increasing pace. Now, USENIX and SAGE have created a conference just for it.

The conference will enable dialogue among peers and experts in our field. We invite you to submit proposals to enhance the invited talks, refereed papers, tutorials, and Birds-of-a-Feather sessions.

Please review this call for participation, make a submission, and join us in creating the first open network administration conference specifically designed to help you.

We look forward to your participation.

Topics

The conference is designed to create an environment allowing network administrators to come together and share ideas and techniques for managing all facets of the networking world. To facilitate these discussions, we encourage you to consider this partial topics list:

The Enterprise

Switching solutions for large LANs
Choosing a network architecture
Network management tools that work
How to integrate multiple vendors in one seamless network
Managing IP address space

The Internet

How to get on the Internet
Firewalls and security integration
Routing in the Internet
Choosing and dealing with an ISP

Remote Access

Creating secure integrated dialup services
DSL, ISDN, and other acronyms: what they mean

The Future

IPv6, and what it means to you
Gigabit and beyond: high speed networking
Quality of Service and ATM
Voice and data integration
Routing vs. Layer-3 switching

What to Submit

The most important segment of the conference is the information dialog it creates. You can do your part by submitting material in one of the following categories.

Invited Talks/Panel Discussions

If you have a presentation that is not suitable for a technical paper submission, please submit a proposal for an invited talk. The proposal should include an extended outline of the talk or panel topic and format include a description of your qualifications to present the topic list the likely participants on the panel conform to the "How and Where ..." instructions below

Acceptance will be based upon the general applicability of the topic and on availability of time in the program.

Refereed Papers

We seek papers relating work of general interest to network administrators, particularly technical papers that reflect hands-on experience or describe real solutions.

Submissions will be judged on the quality of the written submission and whether or not the work advances the art and science of network administration.

A paper submission should:

- contain a short abstract
- include an outline of the paper
- conform to the "How and Where ..." instructions below

If you have a completed paper, you may submit it instead of the abstract and outline.

At least one author of each accepted paper or talk will present the paper during the technical track of the conference. Authors of an accepted refereed paper or invited talk must provide a final paper for publication in the conference proceedings.

Conference proceedings containing all papers will be distributed to attendees and will also be available from USENIX once the conference ends. We also ask that, if possible, copies of presentation slides be made available for wider distribution.

Conference proceedings containing all refereed papers will be distributed to attendees and will also be available from USENIX once the conference ends.

Note that the USENIX organization, as well as most conferences and journals, requires that papers be "unique," i.e., not be submitted to more than one conference or publication. All submissions are held in the strictest confidence prior to publication in the conference proceedings, both as a matter of policy and as protected by the U.S. Copyright Act of 1976.

How and Where to Send Submissions

Please email your submission to
neta-submissions@usenix.org
in any one of the following formats:

- Plain text with no extra markup
- Postscript formated for 8.5" x 11" page
- Microsoft Word
- RTF
- HTML

A cover letter with the following required information in the format below must be included with all submissions:

Authors:	Names and affiliation of all authors
Contact:	Primary contact for the submission
Address:	Contact's full postal address
Phone:	Contact's telephone number
Fax:	Contact's fax number
Email:	Contact's e-mail address
URL:	For all speaker/authors (if available)
Category:	Category of the submission (paper, invited talk, panel)
Title:	Title of the submission
Needs:	Audio-visual requirements for presentation

If you enclose files as an attachment to your submission, please use MIME encoding.

We will acknowledge receipt of a submission by email within one week.

Tutorials

On April 7, there will be full and half-day tutorials in all areas and levels of expertise for network administrators.

If you are interested in presenting a tutorial at the conference, contact the USENIX tutorial coordinator:

Daniel V. Klein
Email: *dvk@usenix.org*
Phone: 412.422.0285
Fax: 412.421.2332

Birds-of-a-Feather (BOF) Sessions

BOF sessions are very informal gatherings of attendees interested in a particular topic. BOFs will be held on Wednesday and Thursday evenings and may be scheduled at the conference or in advance by sending email to *conference@usenix.org.*

Registration Information

Complete program and registration information will be available February 1999 at *http://www.usenix.org/events/neta99.* If you would like to be added to our mailing list, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA USA 92630
Email: *conference@usenix.org*
Phone: 714.588.8649
Fax: 714.588.9706

Call for Papers USENIX

5th Conference on Object-Oriented Technologies and Systems (COOTS '99)

Sponsored by The USENIX Association **Conference Web site: *http://www.usenix.org/events/coots99***

May 3-7, 1999
San Diego, California, USA

Important Due Dates

Paper submissions: *Nov. 6, 1998*
Tutorial submissions: *Nov. 6, 1998*
Notification to authors: *Dec. 16, 1998*
Camera-ready papers: *March 23, 1999*

Conference Organizers

Program Chair
Murthy Devarakonda, *IBM T.J. Watson Research Center*
Program Committee
Ken Arnold, *Sun Microsystems, Inc.*
Rachid Guerraoui, *Swiss Federal Institute of Technology*
Jennifer Hamilton, *Microsoft Corporation*
Doug Lea, *SUNY Oswego*
Gary Leavens, *Iowa State University*
Scott Meyers, *Software Development Consultant*
Ira Pohl, *UC Santa Cruz*
Rajendra Raj, *Morgan Stanley & Company*
Doug Schmidt, *Washington University*
Joe Sventek, *Hewlett-Packard Labs*
Steve Vinoski, *IONA Technologies, Inc.*
Werner Vogels, *Cornell University*
Jim Waldo, *Sun Microsystems*
Yi-Min Wang, *Microsoft Research*
Shalini Yajnik, *Bell Laboratories, Lucent Technologies*
Tutorial Program Chair
Douglas C. Schmidt, *Washington University*

Overview

As the last COOTS before the year 2000, COOTS '99 will focus on "The Object Lessons," our cumulative experiences in building and programming object-oriented systems. We invite you to submit high quality, previously unpublished, original papers on this theme as well as on all topics relating to object-oriented systems.

In addition to experience-centered papers, COOTS '99 accepts papers on a wide range of topics, including but not limited to:

Distributed Objects
Object-oriented systems performance
Security for Distributed Objects
Object services
Mobile objects
Object-oriented design technigues
Component based operating systems
Standard Template Library
Advanced C++ topics/examples
Java and Web programming languages
Container technologies (e.g. Java Beans)
Design patterns
Visual J++ and other development tools
Fault tolerance
New OO programming languages
Object-Oriented database systems
Building distributed applications
Persistent Object Issues
Groupware
Patterns
Major fractures of C++
C++
SmallTalk systems
Commercial toolkits/OBDMS
Platform-independent features of C++

Tutorials

The COOTS conference will begin with two days of tutorials. We expect tutorial topics to include: Distributed object systems (CORBA, DCOM, RMI, etc.), Java and WWW programming languages, framework design, and object-oriented programming languages.

If you are interested in proposing a tutorial, contact the USENIX tutorial coordinator, Dan Klein, by phone at +1.412.422.0285 or by email to **dvk@usenix.org**

Technical Sessions

Two days of technical sessions will follow the tutorials. COOTS emphasizes research and advanced engineering aspects of object technology, focusing on experimental systems research. Conference Proceedings will be published by USENIX and provided free to technical session attendees. An award will be given for the best student paper at the conference.

Advanced Topics Workshop

As usual, the conference will conclude with an Advanced Topics Workshop, where a smaller audience can exchange in-depth technical information on a few position papers. The topic for the ATW will be announced several months before the conference.

What to Submit

Full papers should be 10 to 15 pages (around 5,000-6,000 words). All submissions will be judged on originality, relevance, and correctness.

Each submission must include a cover letter stating the paper title, the contact author, email and regular addresses, and a phone number.

The COOTS conference, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. Additional information and detailed guidelines for submission and examples of extended abstracts can be obtained by sending email to **coots99authors@usenix.org** or by telephoning USENIX at 510.528.8649.

Where to Submit

Please send one copy of a full paper to the program committee via email (Postscript, PDF, or ASCII) to: **coots99papers@usenix.org**. All submissions will be acknowledged.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in February 1999. Please go to the conference Web site:
http://www.usenix.org/events/coots99

If you would like to receive the program materials in print, contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA USA 92630
Phone: 714.588.8649
Fax: 714.588.9706
Email: conference@usenix.org

Announcement and Call for Papers USENIX

USENIX 1999 Annual Technical Conference

For more information about this conference, see http://www.usenix.org/events/usenix99

June 6-11, 1999 **Monterey Conference Center** **Monterey, California, USA**

Important Due Dates

Papers due: December 2, 1998
Author notification: January 20, 1999
Final refereed papers due: April 27, 1999
Final FREENIX papers due: May 11, 1999

Program Committee

Program Chair: Avi Rubin, *AT&T Labs - Research*
Charles Antonelli, *CITI*
Partha Dasgupta, *Arizona State University*
Wu-Chi Feng, *Ohio State University*
Robert Gray, *Boulder Labs*
Peter Honeyman, *USENIX*
Orran Krieger, *IBM*
Anthony LaMarca, *Xerox PARC*
Darrell Long, *University of California, Santa Cruz*
Udi Manber, *University of Arizona*
Gary McGraw, *Reliable Software Technologies*
Yoon-Ho Park, *IBM*
Keith A. Smith, *Harvard University*
Mirjana Spasojevic, *Hewlett-Packard Labs*

Invited Talks Co-Coordinators

Clem Cole, *Digital Equipment Corp.*
John Heidemann, *USC/Information Sciences Institute*

Best Paper Awards

Prizes will be awarded for the best paper and the best paper by a student.

Refereed Papers

Three days of Technical Sessions include one refereed paper track and parallel tracks of Invited Talks and FREENIX Sessions. Refereed papers are published in the Proceedings which are provided to Technical Sessions attendees, along with Invited Talks/FREENIX materials.

The Program Committee seeks original and innovative papers about the applications, architecture, implementation, and performance of modern computing systems. As at all USENIX conferences, papers that analyze problem areas and draw important conclusions from practical experience are especially welcome. Some particularly interesting application topics are:

Availability
Distributed caching and replication
Embedded systems
Extensible operating systems
File systems and storage systems
Interoperability of heterogeneous systems
Mobile code
Mobile computing
Multimedia
New algorithms and applications
Personal digital assistants
Quality of service
Reliability
Security and privacy
Ubiquitous computing and messaging
Web technologies

How to Submit a Paper to the Refereed Track

Authors are required to submit full papers by December 2, 1998.

Please read carefully.

We are looking for mature, full papers. Papers should be 8 to 14 single-spaced 8.5"x11" pages (about 4000-7000 words), not counting figures and references. Papers longer than 14 pages will not be reviewed. Papers so short as to be considered "extended abstracts" will not receive full consideration.

It is imperative that you follow the instructions for submitting a quality paper. Specific questions about submissions may be sent to the program chairman via email to **rubin@usenix.org**. A good paper will demonstrate that the authors:

- are attacking a significant problem
- are familiar with the literature
- have devised an original or clever solution
- if appropriate, have implemented the solution and characterized its performance
- have drawn appropriate conclusions

Note: the USENIX Technical Conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified by January 20, 1999. Some accepted papers will be shepherded by a program committee member through an editorial review process prior to publication in the Proceedings.

How to Send Refereed Paper Submissions

Please follow these two steps:

STEP 1:

Send one copy of your manuscript to the address below. All submissions will be acknowledged.

Preferred method: Postscript submission. See instructions at **www.research.att.com/~rubin/esub.html** You **must** follow the instructions carefully.

Alternate method:

Send 15 hard copies of the manuscript to:

Technical Conference Submission
Aviel D. Rubin
Secure Systems Research Dept.
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932-0971 USA

STEP 2:

In addition, authors must submit the following information (for administrative handling) via email to **usenix99abstracts@research.att.com**. This should be a separate ASCII-only email message.

1. The title of the paper and the names of the authors. (Note: authors' names and

affiliations will be known to the reviewers).
2. The name of one author who will serve as a contact, email and postal addresses, day and evening phone numbers, and a fax number if available.
3. An indication of which, if any, of the authors are full-time students.
4. A short abstract of the paper (100-200 words). This can be the same as the paper's abstract.

FREENIX Track

FREENIX is special track within the conference. USENIX encourages the exchange of information and technologies between commercial product vendors and the free software world, as well as among the various free operating system alternatives themselves, and the FREENIX track has been created to further promote this process. FREENIX attendees may also attend any of the USENIX Conference offerings and informal get-togethers.

FREENIX is also a showcase for the latest developments and interesting applications in the world of freely redistributable software, covering everything from FreeBSD to Linux, GNU to Samba, etc. The FREENIX track attempts to cover the full range of software which is freely redistributable in source code form or provides pointers to where the code can be found on the Internet.

FREENIX Program Committee

Chair: Jordan Hubbard, *FreeBSD*
David Greenman, *FreeBSD*
John Ioannidis
Angelos D. Keromytis, *OpenBSD*
Kirk McKusick, *BSD*
Jason Thorpe, *NetBSD*
Nathan Torkington
Theodore Ts'o, *Linux/GNU*

FREENIX Technical Presentations

We are looking for talks which advance the state-of-the-art of freely redistributable software or otherwise provide useful information to those faced with deploying (and "selling") free software in the field.

Areas of interest include, but are not limited to:

Operating system design
Network design and implementation
File system design
Highly-available systems
Highly-scalable systems
Graphical user interface tools
Desktop metaphors
File and print systems
System management tools
Security
Large scale system management
Interesting deployments of free software
How free software is being developed and managed today

Interesting applications of freely redistributable software might include:

Robotics and automation.
Clustering
Wearable computers
Embedded systems
High-speed networking
Studio graphics
Audio processing

How to Submit a FREENIX Track Presentation

By September 15, 1998, please submit a one page abstract of your proposed presentation, along with a short biography and a comment about your connection with the software involved. This information will be provided to the reviewers of your paper. All topics in the FREENIX track must be about software which is freely redistributable in source code form. Please include pointers to where your code can be found.

Please include the following information for administrative handling:

1. The title of the presentation and the name(s) of the presenters
2. The name of one person who will serve as a contact, along with their email and postal addresses, day and evening phone numbers, and a fax number if available. Please email one copy of the above information (ASCII greatly preferred) to **freenix@usenix.org.**

Tutorial Proposals Welcome

On Sunday, Monday, and Tuesday, June 6-8, USENIX's well-respected tutorial program offers intensive, immediately practical tutorials on topics essential to the use, development, and administration of advanced computing systems. Skilled instructors, who are experts in their topic areas, present both introductory and advanced tutorials covering topics such as:

System and network administration
Java and Perl topics
High availability, scalability, and reliability
Programming tools and program development
Portability and interoperability
System and network security
Client-server application design and development
Sendmail, DNS, and other networking issues
GUI technologies and builders
WWW and CGI technologies
Performance monitoring and tuning
Freely distributable software

If you are interested in presenting a tutorial, contact: Dan Klein, Tutorial Coordinator 412.422.0285, **dvk@usenix.org**

Invited Talks Proposals Welcome

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talks track also may include panel presentations and selections from the best presentations at recent USENIX conferences.

The Invited Talks coordinators welcome suggestions for topics and requests proposals for particular talks. In your proposal state the main focus, include a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to **ITusenix@usenix.org.**

Work-In-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To schedule your short report, send email to **wips99@usenix.org.**

Birds-of-a-Feather Sessions

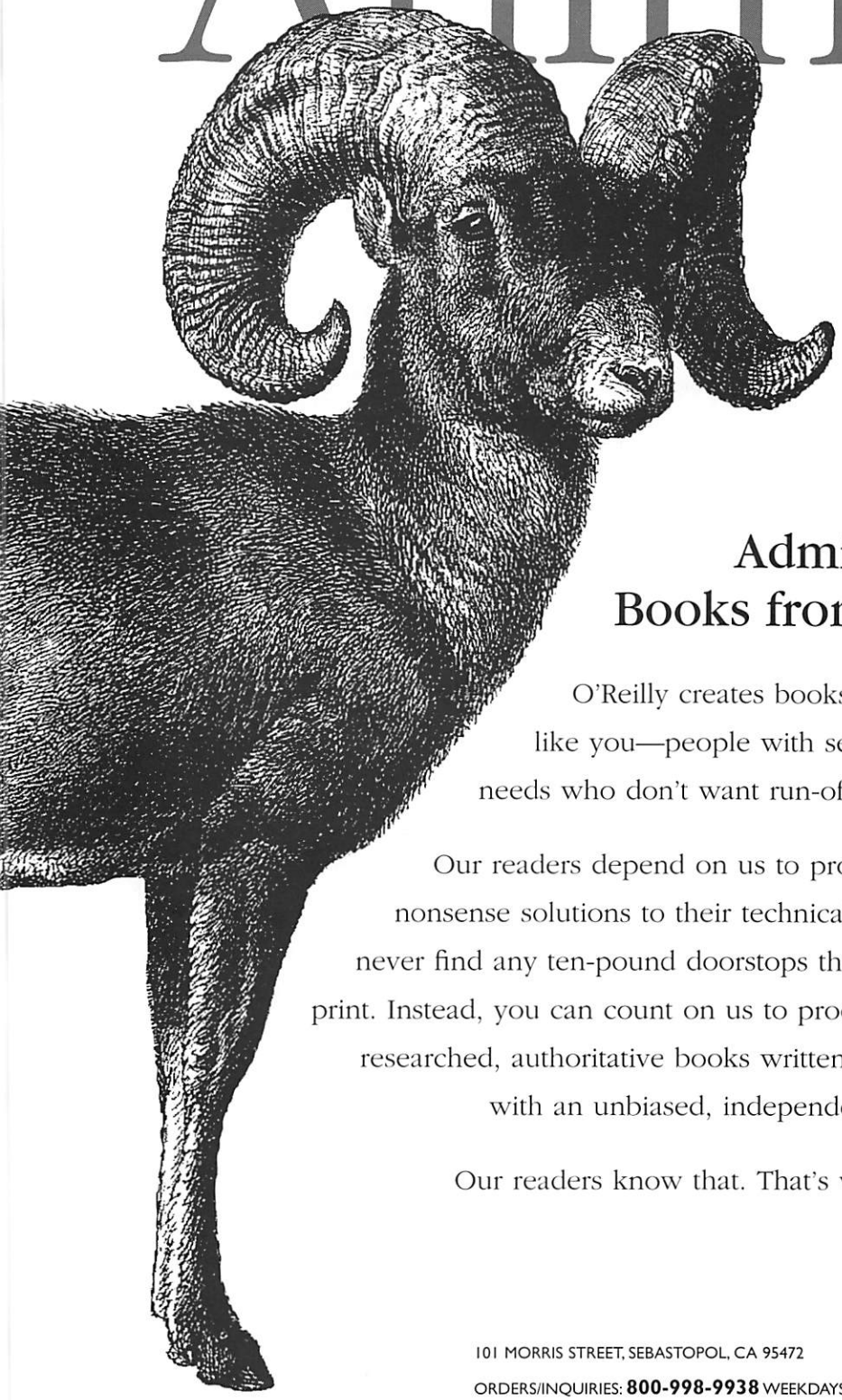
The always popular evening BOFs are very informal, attendee-organized gatherings of persons interested in a particular topic. BOFs may be scheduled at the conference or in advance by contacting the USENIX Conference Office by phone at 714.588.8649 or via email to **conference@usenix.org.**

USENIX Exhibition

In the Exhibition, the emphasis is on serious questions and feedback. Vendors will demonstrate the features and technical innovations which distinguish their products. For more information, including a list of current exhibitors, see **www.usenix.org/events/usenix99/vendors.html.**

Contact:
Cynthia Deno
USENIX Exhibition Coordinator
Telephone: 408.335.9445
Email: **cynthia@usenix.org**

A Different Kind of Animal

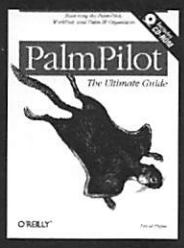


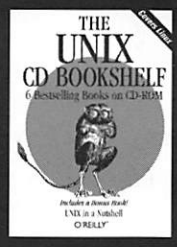
System Administration Books from O'Reilly

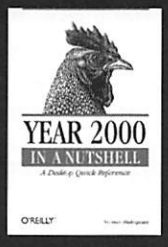
O'Reilly creates books for professionals like you—people with serious information needs who don't want run-of-the-mill answers.

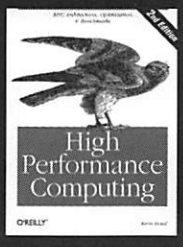
Our readers depend on us to provide reliable, no-nonsense solutions to their technical problems. You'll never find any ten-pound doorstops that are rushed into print. Instead, you can count on us to produce meticulously researched, authoritative books written by the experts—with an unbiased, independent point of view.

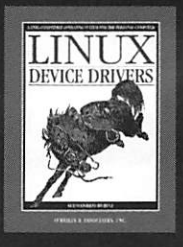
Our readers know that. That's why they trust us.











O'REILLY™
www.oreilly.com

101 MORRIS STREET, SEBASTOPOL, CA 95472

ORDERS/INQUIRIES: **800-998-9938** WEEKDAYS 6AM – 5PM PACIFIC TIME

707-829-0515 • FAX: 707-829-0104

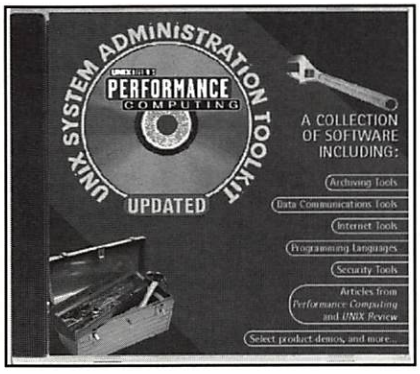
EMAIL FOR OUR CATALOG: **CATALOG@OREILLY.COM**
INCLUDE YOUR NAME AND MAILING ADDRESS

O'REILLY BOOKS ARE ALSO AVAILABLE AT YOUR BOOKSTORE

Please mention code
ALOG
to receive your
20% Usenix discount.

Have All The Tools You Need

Order the *Performance Computing* UNIX System Administration Toolkit CD-ROM



This low-cost comprehensive CD-ROM is packed with tools for the UNIX system administrator. This updated edition includes the latest versions of these tools, thus saving you hours of search and download time. It also includes source code, select software demos from leading solutions providers, and a library of reference material from past issues of *UNIX Review* and *Performance Computing*.

Check out these features*

Programming Tools and Languages
Data Communication Tools
Security Tools
Performance Monitoring Tools
Web Server Software
Web Browsers

*For a complete list of included applications go to *www.performance-computing.com*

Plus...

Select Solutions-oriented Product Demos and a Comprehensive Library of *UNIX Review* and *Performance Computing* Editorial Content Including an Archive of System Management Feature Articles "Perl Advisor" and "Daemons & Dragons" Columns

ALL FOR JUST $39.95!

ORDER FORM

YES! Rush me ____ copies of the ***UNIX System Administration Toolkit CD-ROM*** at $39.95 each!

Quantity	Price	Total
	$39.95	$
	Subtotal	
	Shipping/ Handling	
	Sales Tax	
	TOTAL	$

ORDER TODAY!

Sales Tax: Residents of NY-8%, IL-6.25%, CA-7.5%, TX-7.5%, KS-6.9%, GA-5%

Shipping & Handling: $4 for U.S. and Canada orders; $10 for international orders

Complete this form and return to: *UNIX System Administration Toolkit CD-ROM,* 1601 West 23rd Street, Suite 200, Lawrence, KS 66046. Telephone: 800/444-4881 or 913/841-1631; FAX 913/841-2624; e-mail *orders@mfi.com*

Name _____ Company Name _____

Address _____

City _____ State _____ Zip _____ Country _____

Phone _____ Fax _____

❑ Check ❑ VISA ❑ MasterCard ❑ American Express

Credit Card# _____ Expiration Date _____

Signature _____

The Answer to All Your UNIX Systems Administration Questions

Make Your System Faster, Safer and More Secure!

All you need to keep your system up and running — better than ever!

- Crash Recovery
- Interconnectivity
- Performance Tuning
- Backup
- Web Integration
- NT Integration
- Security
- and more

Get the only magazine devoted 100% to UNIX systems administration — solid, technical information full of ways to improve the performance and extend the capabilities of your system. Regular columns and departments also give you a solid look at important books, new product releases and upgrades, career opportunities, and technical meetings and conferences. Coverage spans a variety of platforms including Solaris, AIX, HP-UX, SCO, Linux and others. If you administer a UNIX system — ***Sys Admin*** can save you and your organization time and money.

Try a FREE issue!
Special Offer for ;Login: readers

You can get a FREE issue of *Sys Admin* sent directly to your home or office! If you like the valuable information, you'll pay only $29 for a full year's subscription (11 additional issues). If not, write "cancel" on the accompanying invoice and owe nothing! The FREE issue is yours to keep!

Save over 51%

Don't Delay!
Start saving time and money today!

Call 1-800-365-2210

Be sure to mention your special ;Login: discount code – 5LGN!

Sys Admin
P.O. Box 59170
Boulder, CO 80322

www.samag.com/sub/login.html

Please allow up to six weeks for delivery of the first issue. Orders outside the U.S. must be prepaid in U.S. funds, drawn on a U.S. bank. CANADA/MEXICO subscriptions are: 1 year - $58; 2 years - $108; 3 years - $153. Overseas subscriptions are: 1 year - $69; 2 years - $129; 3 years - $182. Residents of GA, KS, NM, SC, and TN please add applicable sales tax. Canadian GST included (R#129065819).

57-98

motd

by Rob Kolstad

Rob Kolstad is a long-time USENIX member, having served as chair of several conferences and workshops, director on the Board, and editor of *;login:*. He is also head coach of the USA programming team.

<kolstad@usenix.org>

Winners and . . . Non-winners?

Greetings from sunny Wisconsin where we are wrapping up the final day of the USA Computing Olympiad. We have held the two big-deal final contest rounds and have managed to reduce each of our participants' worth to a four-digit number for purposes of ranking them so we can choose the "very best four" (the "winners") to send to Portugal for the international programming contest to be held in September.

You might notice a certain resemblance here to other, similar procedures in which participants are ranked: job reviews, athletic team tryouts, or even conference paper selection. In each of those activities, there is a limited set of resources (e.g., money, a slot on a team of fixed maximum size, or a slot in a conference program) to be allocated — and there is an oversubscription of sorts for that limited set of resources.

From the coaching (or managing or conference chairing) point of view, we had an embarrassment of riches. Eight participants exceeded our requirements, and thus we were able to use ever more refined evaluation techniques (e.g., arguing loudly) to choose the "very best four." The winners were duly announced and recognized with dandy little trophies whose figurines were made of real brass (a rare find in the basement of the Jensen Plastix trophy store right here in Racine, Wisconsin). These "very best four" were, in some senses, the winners.

That leaves an enigma: what do we call the other eleven participants, some of whom had scores within a few points of the "winners"? Obviously, they must be the "losers" or something like that.

However, finishing in fifth place as a tenth grader in the large field of all pre-college computer programmers is not really what I think of as a tremendously "losing" position. Maybe it's a sort of specifically American thought process to want to embrace only those people (or other entities) that are "the very best" or "number one." Surely there's plenty of evidence to point to this; just tune a TV to any major sports broadcast or open a newspaper to the sports page. Lots of ink for the winners! Those other guys? Well, they're . . . losers.

This convenient classification enables us to partition the participants of many different processes and associate ourselves only with those "winners" who are worthy of our attention and adulation.

I must confess, though, that in the workplace, this is a really tough situation. Mary Smith ends up getting a 6% raise for being a superstar while Joe Black ends up getting a 4% raise for being middle-of-the-road. People too often quickly compare their numerical "scores" to figure out who are the winners and who are the losers. Never mind that Joe Black was already making $18,600 more than Mary. Now he's a loser. He knows it and, if people talk as they often do, Mary knows she is a winner and Joe is a loser. We do spend a lot of time ranking people!

I am a competitive person and I really love winning. Happily, there are enough topics out there that I can choose easy things to try to win. (I am sure my collection of 50,000 electronic typefaces, all neatly categorized in postscript files, is a winner of something.)

I am not so sure, though, that all this competition and adulation for only the very best is the absolute perfect way to run a world. The other eleven participants here are all pretty big winners, too. It surely is a hard thing to communicate that to them, though.

Can we all be winners in our own way? I sure hope so. People do so many good things that should be recognized. I hope we can see beyond the first-place myopia that so pervades many aspects of our society.

Capitalize on your Unix Skills!

UNIX SYSTEM ADMINISTRATION ON ALL LEVELS:

HP-UX, Sun Solaris, AIX

TOOLS AND SKILLS:

- C/C++ Programming
- Perl, CGI, Shell Scripting
- TCP/IP, NFS, NIS
- CGI and Firewalls
- X-Windows, Motif
- SQL

SEND YOUR RESUME TODAY!

IMI Systems Inc. offers experienced computer professionals career opportunities, both salaried and contract, that allow you to build your skills and stay competitive.

imiwest@imisys.com

(800) 828-0180 • (800) 479-4116

www.imiwest.com

Visit our web site or call today to learn more about current opportunities and our consultant benefits package. We offer a variety of career development opportunities including training, tuition reimbursement, relocation assistance and visa sponsorship.

IMI Systems Inc. is an international software development and information technology consulting company. Through a vast network of offices in the United States, Canada and Europe, we serve multinational companies in a variety of industries, including manufacturing, banking, brokerage, insurance, computer and telecommunications.

IMI Systems Inc.

An Olsten Company

Western offices: San Francisco • Palo Alto • Walnut Creek • Sacramento • Irvine • Denver • Colorado Springs • Boulder

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

AUTOMATIC INFORMATION SERVER

If you do not have access to the Web, finger <info@usenix.org> and you will be directed to the catalog which outlines all conferences, activities, and services.

PGP INFORMATION

All correspondence to:
1998 Operational Key
Key ID: 1024/F6F82613 1997/11/21
USENIX 1998 <pgp@usenix.org>
Key fingerprint = 80 6F B5 48 C2 1A B8 45
48 5F F2 38 E6 41 B0 61

1998 Signing Key
Key ID: 1024/4F75A901 1997/11/21
USENIX 1998 Signature
<http://www.usenix.org/pgp/pgpsig.html>
Key fingerprint = 05 FD CF A1 2F 47 00 5C
69 C2 25 E4 66 89 A6 9B

USENIX master key <not-for-email>:
Key ID: 1024/2FEA2EF1 1996/04/08
Key fingerprint = DB A7 50 99 66 E4 8A A9
80 B2 D9 E2 FE DA 00 5E

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are September 29, 1998 and December 9, 1998.

ADVERTISING ACCEPTED

;login: offers an exceptional opportunity to reach 9,000 leading technical professionals worldwide.

Contact: Cynthia Deno
cynthia@usenix.org
408 335 9445

;login:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

I36935